②

Systems
Optimization
Laboratory

Modifying MINOS for Solving the Dual of a Linear Program

by
Eithan Schweitzer

TECHNICAL REPORT SOL 90-11

August 1990

Department of Operations Research
Stanford University
Stanford, CA 94305

SYSTEMS OPTIMIZATION LABORATORY
DEPARTMENT OF OPERATIONS RESEARCH
STANFORD UNIVERSITY
STANFORD, CALIFORNIA 94305-4022

Modifying MINOS for Solving the Dual of a Linear Program

by
Eithan Schweitzer

TECHNICAL REPORT SOL 90-11

August 1990

Research and reproduction of this report were partially supported by the Department of Energy Grant DE-FG03-87ER25028; National Science Foundation Grant DMS-8913089, ECS-8906260; EPRI Contract RP8010-09 and the Office of Naval Research Grant N00014-89-J-1659.

MODIFYING MINOS FOR SOLVING THE DUAL OF A LINEAR PROGRAM

by

Eithan Schweitzer

Abstract:

In solving large-scale linear programs by Bender's decomposition, it can be advantageous to solve the master and the sub problems via their dual problems. In this report I describe the changes I have made in one of the MINOS files, so MINOS could transform a given primal linear program to its dual, solve the dual and in addition, write an MPS file that contains the dual problem.

Introduction

In many situations it is advantageous to solve the dual of a linear program rather than the primal. For example, an approach to solve two stage stochastic linear programs with recourse employs Benders' decomposition (Dantzig and Glynn 1990). Within this algorithm, cuts (necessary conditions for the second stage costs) are added sequentially to the master problem. In the primal version of the master problem, cuts appear as new rows. In each iteration of Benders' decomposition algorithm, when a new row is added, the previous solution of the master problem gets infeasible. However, when solving the dual of the master problem, a cut appears as a column. When adding a new column, the previous solution of the master problem remains feasible and the new optimal solution can be obtained within a few simplex iterations.

In this report I describe the changes I have made in the MINOS file MI35INPT, so MINOS could transform the given primal linear program, specified in an MPS file, to its dual, solve the dual and in addition, write another MPS file which contains the specification of the dual problem.

The dual problem

The primal linear problem, as specified in the MPS file, is given in the following standard form:

$$
\begin{array}{llll}
\text{min (max)} & c^t x & & \\
\text{s.t.} & A x \;\#\; b & & : \pi_1 \\
 & I_1 x \;\leq u & & : \pi_2 \\
 & I_2 x \;\geq l & & : \pi_3 \\
 & I_3 x \;\#\; 0 & & \\
 & -\infty < I_4 x < \infty & &
\end{array}
$$

Where $c^t x$ is the objective function, A is an mxn real matrix of the constraints as given in the COLUMNS section of the MPS file and c is an additional row of A. # denotes relations auch as $\geq$, $\leq$, = as given in the ROWS section. b is the RHS vector. u and l are upper and lower bounds on x, as given in the BOUNDS section, respectively. u and l do

not contain zeros or infinity. The variables that are related to zero or to infinity are associated with I_3, I_4.

The I_i , i=1,...,4 are zero-one matrices, each having n columns and n or less rows. The number of the rows of I_i is the number of the relevant bounds and $I_i(j,k)=1$ if and only if x_k is bounded by the j'th element of the relevant bound vector.

The dual problem is,

$$
\begin{array}{ll}
\text{max (min)} & b^t \pi_1 + u^t \pi_2 + l^t \pi_3 \\
\text{s.t.} & A^t \pi_1 + I_1{}^t \pi_2 + I_2{}^t \pi_3 \ \# \ c \\
 & a_1 \leq \pi_1 \leq a_2 \\
 & \pi_2 \# 0 \\
 & \pi_3 \# 0
\end{array}
$$

Where a_1, a_2, # are determined by the rules of the primal-dual relations. a_1 contains 0 or $-\infty$, a_2 contains 0 or $+\infty$, depending on the types of the rows in the primal problem.

#, a_1 and a_2 are determined according to the following table:

primal:	minimize	maximize
$x_k \geq 0$	$\#_k$ = '$\leq$'	$\#_k$ = '$\geq$'
$x_k \leq 0$	$\#_k$ = '$\geq$'	$\#_k$ = '$\leq$'
$-\infty < x_k < \infty$	$\#_k$ = '='	$\#_k$ = '='
constraint i is '$\geq$'	$a_{1i} = 0$, $a_{2i} = \infty$	$a_{1i} = -\infty$, $a_{2i} = 0$
constraint i is '$\leq$'	$a_{1i} = -\infty$, $a_{2i} = 0$	$a_{1i} = 0$, $a_{2i} = \infty$
constraint i is '='	$a_{1i} = -\infty$, $a_{2i} = \infty$	$a_{1i} = -\infty$, $a_{2i} = \infty$
	$\pi_2 \leq 0$	$\pi_2 \geq 0$
	$\pi_3 \geq 0$	$\pi_3 \leq 0$

I assumed that the primal MPS file has no RANGES section and no initial bounds because they can make the dual problem be too large. However, dealing with these sections can be added later without a lot of efforts.

Creating the dual problem

According to these standard forms of the primal and the dual problems, we have to do the following steps in order to get the dual problem:

1. Change the type of the problem from minimize to maximize or vice versa.
2. Transpose the A matrix.
3. Exchange the places of the b and c vectors.
4. Exchange the names of the constraints and the variables.
5. Set the bounds of π_1.
6. Add u, l, I_1, I_2, to A^t and set the bounds of π_2 and π_3.
7. Set the bounds of the slacks according to the new RHS vector in order to get constraints of the type { $Ax+Is = 0$, $lb \le x,s \le ub$ } that will fit the MINOS' data structure.

All the modifications I made are written in a file named DU35INPT which contains the original MI35INPT file with the modifications. The new co-subroutines I have written are in a file named DUTRANSPOSE, so, all one has to do in order to execute the modified MINOS is to link DU35INPT and DUTRANSPOSE (which can always be combined together) with the rest of the MINOS code, but without MI35INPT.

DUTRANSPOSE file

This file contains modular subroutines that deal with transposing A and exchanging the names of the rows and the columns of the primal problem.

Subroutine TRANSPOS

This subroutine transposes the A matrix and returns A^t.

Specification:

```
SUBROUTINE TRANSPOS(A,HA,KA,KA2,NE,NKA)
INTEGER*2        NE
DOUBLE PRECISION A(NE)
INTEGER*2        HA(NE), KA2(NE)
INTEGER          MCOLS
INTEGER          KA(NKA)
```

Parameters:

A(*) - (input, output) The non zero elements of the constraints matrix, ordered column-wise. At the end of TRANSPOS it contains the same elements ordered row-wise, so, in fact, it contains the non zero elements of A^t ordered column-wise.

HA(*) - (input, output) The row indices associates with A at the beginning and with A^t at the end.

KA(*) - (input, output) KA(j) points the start of column j in the arrays A and HA.

KA2(*) - (temporary) Is a far and unused part of Z(*) and during the execution of TRANSPOS it contains the column indices associates with A or A^t as required.

NE - (input) The number of non-zero elements in the A matrix, which has the length of A, HA and KA2.

NKA - (input) The length of KA.

TRANSPOS uses the subroutine HEAPSORT which sorts A, HA and KA2 together using HA as the key vector of the sorting. HEAPSORT uses the subroutines PUSHDOWN and SWAP.

Subroutine EXNAMES

This subroutine exchanges the names of the rows and the columns after A has been transposed.

Specification:

```
      SUBROUTINE EXNAMES(M,N,IOBJ,IOBJOLD,
     $           KEYNAM,LENH,NAME1R,NAME2R,HRTYPE,MOLD,
     $           NAME1C,NAME2C,NOLD,TNCOL,THRTYP)
      INTEGER    KEYNAM(LENH)
      INTEGER    NAME1R(MOLD), NAME2R(MOLD), TNCOL(NOLD*2)
      INTEGER*2  HRTYPE(MOLD)
      DOUBLE PRECISION  THRTYP(MOLD)
      INTEGER    NAME1C(NOLD), NAME2C(NOLD)
```

Parameters:

M - (input) The number of the rows in A^t.

N - (input) The number of the columns in A^t.

IOBJ - (input) The row number of the dual objective function.

IOBJOLD - (input) the row number of the primal objective function in the original A matrix.

KEYNAM(*) - (input, output) A hash table of indices to the row names.

LENH - (input) The length of KEYNAM.

NAME1R(*), NAME2R(*) - (input,output) The row names.

HRTYPE(*) - (input) The types of the primal rows.

MOLD - (input) The number of rows in the original A matrix.

NAME1C(*), NAME2C(*) - (input, output) The columns names.

NOLD - (input) The number of the columns in the original A matrix.

TNCOL - (temporary) A far and unused part of Z(*) that will hold the names of the columns while exchanging places.

THRTYP - (output) A far and unused part of Z(*) that will hold the original row types.

Subroutine DUALIST

This subroutine finds a name in the column's names vector.

Specification:

```
SUBROUTINE DUALIST (N,M,ID1,ID2, NAME1C,NAME2C,IP,FOUND)
INTEGER NAME1C(N), NAME2C(N)
```

Parameters:

N - (input) The number of the columns.

ID1, ID2 - (input) The name to search for.

NAME1C(*), NAME2C(*) - the column names vector.

IP - (output) The place of ID1, ID2 in NAME1C, NAME2C.

FOUND - (output) .TRUE. if and only if ID1,ID2 were found in NAME1C, NAME2C.

Subroutine DUIXTOS

Converts an integer into a numeric string.

Specifications:

```
SUBROUTINE DUIXTOC(I,CH)
CHARACTER*4   CH
```

Parameters:

I - (input) An integer, for example: 257.

CH - (output) The string, for example: '0257'.

DUIXTOC uses DUITOC to convert a number into a character.

DU35INPT file

The original MI35INPT contains the following subroutines: M3INPT, M3MPSA, M3MPSB, M3MPSC, M3READ and M2CORE.

I put some changes in M3INPT, M3MPSA, M3MPSB and splitted M3MPSC to two subroutines: DUBOUNDS and M3MPSC. The new M3MPSC is exactly the same as it is in the original M3MPSC. M3READ and M2CORE are unchanged.

Subroutine M3INPT

M3INPT, originally, is the subroutine that reads the MPS file. The new M3INPT does the same, but, in addition, it changes the data structures of MINOS so they will contain the dual problem. In addition, M3INPT creates a new file, 'minos.dul', which will contain the dual problem in MPS format.

The additional operations in M3INPT are the following:

COMMON /DUFILE/ IDUAL

IDUAL is the logic name of 'minos.dul' file and is common to M3INPT, M3MPSA, M3MPSB and DUBOUNDS.

At the beginning of M3INPT, the IDUAL file is opened as new. Then an additional space for the dual problem must be determined. MELEMS, the maximal number of elements in A can grow to MELEMS+2*MCOLS, where MCOLS is the maximal number of columns as specified in the SPEC file. In the dual problem the number of the rows is exactly the number of the original variables, so MROWS <-- MCOLS, but MCOLS, which should get the value of MROWS, can grow to MROWS+2*MCOLS.

After the space for the dual problem has been determined, M2CORE will allocate enough space for the relevant vectors in Z(*). Then, M3MPSA can be called to read the rows from the MPS file. After the rows are inputted and M, the number of the rows, is known, M3MPSB can be called to read columns and the RHS and to exchange rows and columns.

M3MPSB is called with three more parameters: Z, NWCORE and LPI. LPI is the place in Z(*) that contains the extra memory needed in TRANSPOS and in EXNAMES.

After M3MPSB, M and N are known so we can call DUBOUNDS to read the bounds from the MPS file, add the relevant elements to A, set bounds to the dual variables and to the slacks and write the 'minos.dul' file.

At the end of DUBOUNDS, M remains the same, N and NE are known and we can give MCOLS, MROWS and MELEMS their exact values. At that point, the data are structured as if the original MINOS had read an MPS file that specifies the dual problem. So, from now on, we can proceed as in the original M3INPT.

Subroutine M3MPSA

Specification:

```
      SUBROUTINE M3MPSA( MROWS, MCOLS, MELMS, LENH, NCOLL, M,
     $                   NN, NNCON, KEY, NCARD,
     $                    HRTYPE, NAME1R, NAME2R, KEYNAM )

      IMPLICIT           REAL*8(A-H,O-Z)
      CHARACTER*4          KEY
      DIMENSION           KEY(3), NCARD(6)
      INTEGER*2          HRTYPE(MROWS)
      INTEGER           KEYNAM(LENH)
      INTEGER           NAME1R(MROWS), NAME2R(MROWS)
```

All the parameters have exactly the same meaning as in the original M3MPSA.

M3MPSA inputs the NAME and ROWS sections of the MPS file. The only chnages in M3MPSA are:

1. MINIMZ <--- -MINIMZ, where MINIMZ = 1 if the problem is a minimization one, and -1 if it is a maximization problem. The content of MINMAX is also changed from 'min' to 'max' or vice versa, as required. This change comes after operation number 15.

2. When knowing the name of the objective function after operation 70, that name is given to the RHS.

Subroutine M3MPSB

M3MPSB inputs the COLUMNS, RHS and RANGERS sections of the MPS file.

Specification:

```
 SUBROUTINE M3MPSB( Z,NWCORE,IEXPLACE,MCOLS, MELMS, LENH, NCOLL,
 $              M, N, NB, NE, NKA,
 $              NN, NNCON, NNJAC, NNOBJ, NJAC, KEY, NCARD,
 $               HRTYPE, NAME1R, NAME2R, KEYNAM,
 $              KA, HA, A, BL, BU, KB, NAME1C, NAME2C)

  IMPLICIT         REAL*8(A-H,O-Z)
 CHARACTER*4        KEY
  DIMENSION         KEY(3), NCARD(6)
  INTEGER*2         HRTYPE(M), HA(MELMS)
  INTEGER          KA(NKA),  KB(M), KEYNAM(LENH)
  INTEGER          NAME1R(M),    NAME2R(M)
 INTEGER          NAME1C(MCOLS), NAME2C(MCOLS)
  DOUBLE PRECISION   A(MELMS), BL(NB), BU(NB), Z(NWCORE)
```

All the parameters have exactly the same meaning as in the original M3MPSB. The following parameters have been added to M3MPSB's parameters list:

Z, NWCORE - (input) Are in the same connotation as MINOS uses.

IEXPLACE - (input) Is the place in Z(*) that contains the extra memory needed for TRANSPOS and for EXNAMES.

The changes in M3MPSB are the following:

1. Insert zeros to the first column of A, before A is read, in order to keep room for the objective of the dual. This change is placed at the beginning of M3MPSB around operation number 190.

2. When M3MPSB finishes to read the COLUMNS section, A is known. So, after operation number 400, M3MPSB transposes the A matrix by using TRANSPOS, save the values of M and N in MOLD and NOLD respectively and exchanges the values of M and N. Then, it removes the original objective vector from A to Z(IEXPLACE+1) which is Z(LPI+1) and sets IOBJ to be 1. IOBJOLD is the number of the row that originally held the objective vector. After that, EXNAMES is called to exchange the names of the rows and the columns and then, the bounds of π_i are set according to the original type of the rows that

are held in Z(IEXPLACE+M+3*NOLD+3) and were put there by EXNAMES. All of that is done between operations 400 and 407.

3. Input the RHS of the primal problem, give its name to the dual objective vector and put its values in IOBJ row of A, which contained zeros until now. This is done between operations number 410 and 460.

4. The treatment of the phantom elements and the slacks is moved to DUBOUNDS because the bounds have to be known first.

The rest of M3MPSB, i.e. inputing the RANGES, was not treated and remains the same.

Subroutine DUBOUNDS

This subroutine is an expanded part of M3MPSC. M3MPSC was divided to two subroutines, the first is named DUBOUNDS and the name of the second remained M3MPSC. From operation 700 and on M3MPSC remains the same and also has the same parameters. The operations that come before number 700 are moved to DUBOUNDS.

Specification:

```
      SUBROUTINE DUBOUNDS( M, N, NE, NKA, MELMS, KEY, NCARD,
     $                    MCOLS, NB, A, HA, KA, HRTYPE,
     $                    NAME1C, NAME2C, NAME1R, NAME2R,
     $                    BL, BU, VRHS )

      IMPLICIT          REAL*8(A-H,O-Z)
      CHARACTER*4       KEY
      DIMENSION         KEY(3), NCARD(6)
      INTEGER*2         HRTYPE(M), HA(MELMS)
      INTEGER           KA(NKA), NAME1R(M), NAME2R(M)
      INTEGER           NAME1C(MCOLS), NAME2C(MCOLS)
      DOUBLE PRECISION  A(MELMS), BL(NB), BU(NB), VRHS(M)
```

All the parameters have exactly the same meaning as in the original M3INPT except of VRHS which is the vector (a far part of Z) that holds the dual RHS which was the primal objective and was read in M3MPSB. BL and BU are the bounds for the variables - for the dual variables in this implementation.

DUBOUNDS does the following:

1. Sets default rows type to $\leq$ if the dual problem is a minimization problem and to $\geq$ if the dual problem is a minimization problem. It also sets the type of the objective vector to be 'N' (around operation number 100)

2. Reads the bound of the primal variables from the MPS file. If the primal variable is related to zero or to plus or minus infinity, DUBOUNDS sets the relevant row's type according to the primal-dual transformation rules (operations 690-696). If the primal variable is related to a non-zero finite bound, DUBOUNDS add a new column to A and sets the bounds of the related dual variable to be ≥ 0 or ≤ 0 according to the primal-dual transformation rules (operations 660-680).

3. Prints the dual MPS file to 'minos.dul' (900).

4. Inserts Phantom columns for cycling algorithms (5002).

5. Initializes bounds for the slacks according to the type of their rows (5006).

From this point there are no more changes in MINOS.

References

Aho A.V., Hopcroft J.E. and Ullman J.D., *"Data Structures and Algorithms"*, Addison - Wesley Publishing Company (1983).

Dantzig G.B., *"Linear Programming and Extensions"*, Princeton University Press, Princeton (1963).

Dantzig G.B. and Glynn P.W., *"Parallel Processors for Planning Under Uncertainty"*, Annals of Operations Research 22, 1-21 (1990).

Entriken R., *"Using MINOS as a Subroutine for Decomposition"*, Technical report SOL 87-7, Stanford University (1987).

Entriken R. and Infanger G., *"Decomposition and Importance Sampling For Stochastic Linear Models"*, Energy, The International Journal, July-August 1990.

Infanger G., *"Monte Carlo (Importance) Sampling Within a Benders' Decomposition Algorithm for Stochastic Linear Programs"*, Technical report SOL 89-13R, Department of Operations Research, Stanford University, submitted for publication (1990).

Murtagh B.A. and Saunders M.A., *"MINOS 5.1 User's Guide"*, Technical report SOL 83-20R, Stanford University (1983).

DUTRANSPOSE.FOR - file listing

```
      SUBROUTINE TRANSPOS(A,HA,KA,KA2,NE,NKA)
      INTEGER*2        NE
      DOUBLE PRECISION A(NE)
      INTEGER*2        HA(NE), KA2(NE)
      INTEGER          MCOLS
      INTEGER          KA(NKA)
*     -----------------------------------------------
*      TRANSPOS TRANSPOSES THE MATRIX A AND CHANGES
*      HA AND KA RESPECTIVELY.
*      KA2 IS AN ARRAY OF HA'S TYPE AND SHOULD BE DEFINED
*      BEFORE CALLING TRANSPOS. KA2 CAN BE A FAR AND UNUSED
*      PART OF Z.
*     -----------------------------------------------
      K = 1
      DO 40 I=1,NE
   10  KA2(I) = K
        IF (I .LT. KA(K+1) .OR. I .GE. NE) GO TO 40
        K = K+1
        GO TO 10
   40 CONTINUE
      CALL HEAPSORT(A,HA,KA2,NE)
      K = 1
      KA(1) = 1
      DO 60 I=2,NE
        IF (HA(I) .EQ. K) GO TO 60
        K = K+1
        KA(K) = I
   60 CONTINUE
      KA(K+1) = NE+1
      DO 70 I=1,NE
        HA(I) = KA2(I)
   70 CONTINUE
      RETURN
      END

      SUBROUTINE HEAPSORT(A,HA,KA2,NE)
      INTEGER*2        NE
      DOUBLE PRECISION A(NE)
      INTEGER*2        HA(NE),KA2(NE)
*     heapsort algorithm to sort A,HA and KA2 using HA as
*      a common key.
*     HEAPSORT uses PUSHDOWN and SWAP.
      K = NE/2
      DO 10 I=K,1,-1
        CALL PUSHDOWN(I,NE,A,HA,KA2,NE)
   10 CONTINUE
      DO 20 I=NE,2,-1
        CALL SWAP(1,I,A,HA,KA2,NE)
        CALL PUSHDOWN(1,I-1,A,HA,KA2,NE)
   20 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE PUSHDOWN(IFIRST,ILAST,A,HA,KA2,NE)
       INTEGER*2 IFIRST,ILAST,NE
       DOUBLE PRECISION A(NE)
       INTEGER*2       HA(NE),KA2(NE)
       INTEGER         R
       R = IFIRST
    10 IF (R .GT. ILAST/2) GO TO 90
       IF (R. EQ. ILAST/2) GO TO 20
       IF (HA(R) .LE. HA(2*R) .AND. HA(2*R) .GT. HA(2*R+1)) GO TO 30
       IF (HA(R) .LE. HA(2*R+1) .AND. HA(2*R+1) .GE. HA(2*R)) GO TO 40
       R = ILAST
       GO TO 10
    20 IF (HA(R) .GT. HA(2*R)) GO TO 25
       CALL SWAP(R,2*R,A,HA,KA2,NE)
    25 R = ILAST
       GO TO 10
    30 CALL SWAP(R,2*R,A,HA,KA2,NE)
       R = 2*R
       GO TO 10
    40 CALL SWAP(R,2*R+1,A,HA,KA2,NE)
       R = 2*R+1
       GO TO 10
    90 RETURN
       END

      SUBROUTINE SWAP(I,J,A,HA,KA2,NE)
       INTEGER*2       I,J,NE
      DOUBLE PRECISION A(NE)
      INTEGER*2       HA(NE),KA2(NE)
      IR = HA(I)
      HA(I) = HA(J)
      HA(J) = IR
      IR = KA2(I)
      KA2(I) = KA2(J)
      KA2(J) = IR
      TR = A(I)
      A(I) = A(J)
      A(J) = TR
      RETURN
      END


      SUBROUTINE EXNAMES(M,N,IOBJ,IOBJOLD,
     $              KEYNAM,LENH,NAME1R,NAME2R,HRTYPE,MOLD,
     $              NAME1C,NAME2C,NOLD,TNCOL,THRTYP)
      INTEGER    KEYNAM(LENH)
      INTEGER    NAME1R(MOLD), NAME2R(MOLD), TNCOL(NOLD*2)
      INTEGER*2  HRTYPE(MOLD)
      DOUBLE PRECISION  THRTYP(MOLD)
      INTEGER    NAME1C(NOLD), NAME2C(NOLD)
*     Excahnge the names of the rows and the columns.
      LOGICAL    FOUND

*     copy columns names to TNCOL.
      DO 10 I=0,NOLD-1
       TNCOL(2*I) = NAME1C(I+1)
```

```
      TNCOL(2*I+1) = NAME2C(I+1)
   10 CONTINUE
      DO 20 I=1,N
        NAME1C(I) = '    '
        NAME2C(I) = '    '
   20 CONTINUE
*     new list of columns names
      DO 30 I=1,LENH
        IF (KEYNAM(I) .EQ. 0) GO TO 30
        IA = KEYNAM(I)
        IF (IA .EQ. IOBJOLD) GO TO 30
        J = IA
        IF (IA .GT. IOBJOLD) J = IA-1
        THRTYP(J) = HRTYPE(IA)
        NAME1C(J) = NAME1R(IA)
        NAME2C(J) = NAME2R(IA)
   30 CONTINUE
*     erase old rows data structure
      DO 40 I=1,LENH
        KEYNAM(I) = 0
   40 CONTINUE
      DO 45 I=1,N
        NAME1R(I) = '    '
        NAME2R(I) = '    '
        HRTYPE(I) = 0
   45 CONTINUE
*     insert new rows to data structure
      DO 55 I=1,M
        ID1 = TNCOL(2*I-2)
        ID2 = TNCOL(2*I-1)
        CALL M1HASH(LENH,M,N,ID1,ID2,2,KEYNAM,NAME1R,NAME2R,IA,FOUND)
        IF (FOUND) GO TO 50
          KEYNAM(IA) = I
          NAME1R(I) = ID1
          NAME2R(I) = ID2
        GO TO 55
   50   WRITE(IDUAL,'(4X,A22,5I,2X,2A4)') ' ERROR IN ETERING ROW ',
     $               I,ID1,ID2
   55 CONTINUE

      RETURN
      END

      SUBROUTINE DUALIST(N,ID1,ID2,NAME1C,NAME2C,IP,FOUND)
      INTEGER  NAME1C(N), NAME2C(N)
      LOGICAL  FOUND
*     find ID1,ID2 in NAME1C,NAME2C
      FOUND = .FALSE.
      IP = 0
  310 IP = IP+1
      IF (IP .GT. N) GO TO 350
      IF (ID1 .NE. NAME1C(IP) .OR. ID2 .NE. NAME2C(IP)) GO TO 310
      FOUND = .TRUE.
  350 RETURN
      END
```

```
      SUBROUTINE DUITOC(I,C)
      CHARACTER C
*     Transform I to a character.
      IF (I .EQ. 0) C = '0'
      IF (I .EQ. 1) C = '1'
      IF (I .EQ. 2) C = '2'
      IF (I .EQ. 3) C = '3'
      IF (I .EQ. 4) C = '4'
      IF (I .EQ. 5) C = '5'
      IF (I .EQ. 6) C = '6'
      IF (I .EQ. 7) C = '7'
      IF (I .EQ. 8) C = '8'
      IF (I .EQ. 9) C = '9'
      RETURN
      END

      SUBROUTINE DUIXTOS(I,CH)
      CHARACTER*4 CH
      CHARACTER C
*     Transform I to a string[4].
      K = MOD(I,10)
      J = I/10
      CALL DUITOC(K,C)
      CH = C
      DO 5 IK=1,3
        K = MOD(J,10)
        J = J/10
        CALL DUITOC(K,C)
        CH = C//CH
    5 CONTINUE
      RETURN
      END
```

DU35INPT.FOR - file listing

```
*************************************************************************
*
*   File DU35INPT FORTRAN.
*
*   M3INPT  M3MPSA  M3MPSB  DUBOUNDS  M3MPSC  M3READ
*
*   M2CORE
*
*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

      SUBROUTINE M3INPT( Z, NWCORE )
      IMPLICIT           REAL*8(A-H,O-Z)
      DOUBLE PRECISION   Z(NWCORE)

*     ------------------------------------------------------------------
*     M3INPT  inputs constraint data in MPS format,
*     transfers the problem to its dual and sets up
*     various quantities as follows:
*
*     M, N, NE   are the number of rows, columns and elements in A.
*
*     IOBJ       is the row number for the linear objective (if any).
*                It must come after any nonlinear rows.
*                IOBJ = 0 if there is no linear objective.
*
*     A, HA, KA  is the matrix A stored in Z at locations LA, LHA, LKA.
*     BL, BU     are the bounds  stored in Z at locations LBL, LBU.
*     HS, XN     are states and values   stored in Z at   LHS, LXN.
*
*     HS(J)      is set to  0, 1  to indicate a plausible initial state
*                (at LO or UP bnd) for each variable J  (J = 1 to NB).
*                If CRASH is to be used, i.e., CRASH OPTION gt 0 and
*                if no basis file will be supplied, the INITIAL BOUNDS
*                set may initialize HS(J) as follows to assist CRASH:
*
*     -1      if column or row J is likely to be in the optimal basis,
*      4      if column J is likely to be nonbasic at its lower bound,
*      5      if column J is likely to be nonbasic at its upper bound,
*      2      if column or row J should initially be superbasic,
*      0 or 1 otherwise.
*
*     XN(J)      is a corresponding set of initial values.
*                Safeguards are applied later by M4CHEK, so the
*                values of HS and XN are not desperately critical.
*
*     The arrays NAME, MOBJ, MRHS, MRNG, MBND are loaded with the
*     appropriate names in 2A4 format.  (See subroutine M1CHAR.)
*
*     NAME and the ROW and COLUMN names are output to the SCRATCH file
*     in 2A4 format, for use in basis files and the printed solution.
*
*     ------------------------------------------------------------------

      COMMON    /M1FILE/ IREAD,IPRINT,ISUMM
      COMMON    /M2FILE/ IBACK,IDUMP,ILOAD,IMPS,INEWB,INSRT,
```

```
$              IOLDB,IPNCH,IPROB,ISCR,ISOLN,ISPECS,IREPRT
 COMMON    /M2LEN / MROWS ,MCOLS ,MELMS
 COMMON    /M2MAPA/ NE   ,NKA  ,LA   ,LHA  ,LKA
 COMMON    /M2MAPZ/ MAXW ,MAXZ
 COMMON    /M3LEN / M    ,N    ,NB   ,NSCL
 COMMON    /M3LOC / LASCAL,LBL  ,LBU  ,LBBL ,LBBU ,
$              LHRTYP,LHS  ,LKB
 COMMON    /M3MPS1/ LNAM1C,LNAM1R,LNAM2C,LNAM2R,LKEYNM
 COMMON    /M3MPS2/ LCNAM1,LRNAM1,LCNAM2,LRNAM2,LXS,LXL,LFREE
 COMMON    /M3MPS3/ AIJTOL,BSTRUC(2),MLST,MER,
$              AIJMIN,AIJMAX,NA0,LINE,IER(20)
 COMMON    /M5LEN / MAXR ,MAXS ,MBS  ,NN   ,NN0  ,NR   ,NX
 COMMON    /M5LOC / LPI  ,LPI2 ,LW   ,LW2  ,
$              LX   ,LX2  ,LY   ,LY2  ,
$              LGSUB ,LGSUB2,LGRD ,LGRD2 ,
$              LR   ,LRG  ,LRG2 ,LXN
 COMMON    /M5LOG1/ IDEBUG,IERR,LPRINT
 COMMON    /M5PRC / NPARPR,NMULPR,KPRC,NEWSB
 COMMON    /M7LEN / FOBJ ,FOBJ2 ,NNOBJ ,NNOBJ0
 COMMON    /M7LOC / LGOBJ ,LGOBJ2
 COMMON    /M8LEN / NJAC ,NNCON ,NNCON0,NNJAC
 COMMON    /M8LOC / LFCON ,LFCON2,LFDIF ,LFDIF2,LFOLD ,
$              LBLSLK,LBUSLK,LXLAM ,LRHS ,
$              LGCON ,LGCON2,LXDIF ,LXOLD
 COMMON    /DUFILE/ IDUAL
*    -----------------------------------------------------------

 INTRINSIC       MAX, MOD

 CHARACTER*4     KEY
 DIMENSION       KEY(3), NCARD(6)
 CHARACTER*4     LE, LND
 DATA            LE, LND    /'E   ','ND  '/

 IDUAL = 18
 OPEN(UNIT=IDUAL, FILE='MINOS.DUL', STATUS='NEW')

*    (dual) Determine space for the dual problem

 MELEMS = MELEMS+(MCOLS*2)
 I = MROWS
 K = MCOLS
 MROWS = MCOLS
 MCOLS = I+(K*2)

*    Start.  We may come back here to try again with more workspace.

10 IERR   = 0
 NCOLL  = 0
 KEY(1) = LE
 KEY(2) = LE
 KEY(3) = LE
 DO 30 I  = 1, 6
30 NCARD(I) = 0

*    Find a prime number for the length of the row hash table.

 LENH   = MROWS*2
```

```
      LENH   = MAX( LENH, 100 )
      LENH   = (LENH/2)*2 - 1
      K      = LENH/20 + 6

  100 K      = K + 1
      LENH   = LENH + 2
      DO 120 I = 3, K, 2
         IF (MOD(LENH,I) .EQ. 0) GO TO 100
  120 CONTINUE

      CALL M2CORE( 2, MINCOR )
      IF (MAXZ .LT. MINCOR) GO TO 600

*     ---------------------------------------------------------
*     Input ROWS.
*     ---------------------------------------------------------
      CALL M3MPSA( MROWS, MCOLS, MELMS, LENH, NCOLL, M,
     $             NN, NNCON, KEY, NCARD,
     $             Z(LHRTYP), Z(LNAM1R), Z(LNAM2R), Z(LKEYNM) )
      IF (IERR .EQ. 40) GO TO 400
      IF (IERR .EQ. 41) GO TO 500

*     ---------------------------------------------------------
*     M  is now known.
*     Input COLUMNS, RHS, RANGES.
*     (dual) There are additional parameters passed to M3MPSB
*     ---------------------------------------------------------
      MROWS  = M
      CALL M3MPSB( Z,NWCORE,LPI,MCOLS, MELMS, LENH, NCOLL,
     $             M, N, NB, NE, NKA,
     $             NN, NNCON, NNJAC, NNOBJ, NJAC, KEY, NCARD,
     $             Z(LHRTYP), Z(LNAM1R), Z(LNAM2R), Z(LKEYNM),
     $             Z(LKA), Z(LHA), Z(LA), Z(LBL), Z(LBU),
     $             Z(LKB), Z(LNAM1C), Z(LNAM2C) )
      IF (IERR .EQ. 40) GO TO 400
      IF (IERR .EQ. 41) GO TO 510
      NB = M + MCOLS
*     ---------------------------------------------------
*     (dual) Input BOUNDS. DUBOUNDS is the first half of the
*     original M3MPSC that deals with bounds' input.
*     ---------------------------------------------------
      CALL DUBOUNDS( M, N, NE, NKA, MELMS, KEY, NCARD,
     $             MCOLS, NB, Z(LA), Z(LHA), Z(LKA), Z(LHRTYP),
     $             Z(LNAM1C), Z(LNAM2C), Z(LNAM1R), Z(LNAM2R),
     $             Z(LBL), Z(LBU), Z(LPI+1) )

*     ---------------------------------------------------
*     N  and  NE  are now known.
*     ---------------------------------------------------
      MCOLS  = N
      MROWS  = M
      MELMS  = NE
      NP1    = N + 1
      NB     = N + M
      IF (MAXS .GT. NP1) MAXS = NP1
      IF (MAXR .GT. NP1) MAXR = NP1
      IF (NN   .GE. NB ) NN   = NB
```

```
*   (dual) M3MPSC is the second half of the original M3MPSC
*   that deal with initial bounds, a feature that is not treated
*   in the dual case.

    CALL M3MPSC( M, N, NB, NE, NS,
   $          KEY, NCARD,
   $          Z(LBL), Z(LBU), Z(LHS),
   $          Z(LXN), Z(LNAM1C), Z(LNAM2C) )

*   ------------------------------------------------------------
*   Fiddle with PARTIAL PRICE parameter to avoid foolish values.
*   ------------------------------------------------------------
    IF (LPRINT .GT. 0) WRITE(IPRINT, 1300) LENH,NCOLL
    IF (NA0    .GT. 0) WRITE(IPRINT, 1320) NA0
    IF (NNCON  .GT. 0) WRITE(IPRINT, 1350) NJAC
    IF (NN     .GT. 0  .OR.  NCARD(6) .GT. 0)
   $             WRITE(IPRINT, 1400) NCARD(6),NS
    IF (NPARPR .GT. MAX0(N,M)) NPARPR = MIN0(N,M)
    NPR1  = N/NPARPR
    NPR2  = M/NPARPR
    IF (NPARPR .GT. 1) WRITE(IPRINT, 1500) NPR1,NPR2

*   ------------------------------------------------------------
*   Compress storage, now that we know the size of everything.
*   ------------------------------------------------------------

*   Save current positions of  BL, BU, etc.

    KHA   = LHA
    KKA   = LKA
    KBL   = LBL
    KBU   = LBU
    KHS   = LHS
    KXN   = LXN

*   Redefine addresses in  Z  in terms of the known dimensions.

    CALL M2CORE( 3, MINCOR )
    IF (MAXZ .LT. MINCOR) GO TO 800

*   Move BL, BU, etc. into their final positions.

    CALL HCOPY( NE, Z(KHA),   Z(LHA)  )
    CALL ICOPY( NKA,Z(KKA),   Z(LKA)  )
    CALL DCOPY( NB, Z(KBL),1, Z(LBL),1 )
    CALL DCOPY( NB, Z(KBU),1, Z(LBU),1 )
    CALL HCOPY( NB, Z(KHS),   Z(LHS)  )
    CALL DCOPY( NB, Z(KXN),1, Z(LXN),1 )
    GO TO 900

*   --------------------------
*   Fatal error in MPS file.
*   --------------------------
400 CALL M1PAGE( 2 )
    WRITE(IPRINT, 1100)
    WRITE(ISUMM , 1100)
    GO TO 700
```

```
*   --------------------------
*   Too many rows.
*   --------------------------
 500 MROWS  = M
     GO TO 520

*   ---------------------------
*   Too many columns or elements.
*   ---------------------------
 510 MCOLS  = N
     MELMS  = NE

*   Try again.

 520 IF (IMPS .EQ. IREAD) GO TO 600
     REWIND ISCR
     REWIND IMPS
     GO TO 10

*   -------------------------------
*   Not enough core to read MPS file.
*   -------------------------------
 600 CALL M1PAGE(2)
     WRITE(IPRINT, 1110)
     WRITE(ISUMM , 1110)
     IERR   = 41

*   --------------------------------
*   Flush MPS file to the ENDATA card.
*   --------------------------------
 700 IF (IMPS .NE. IREAD) GO TO 900
     DO 750 IDUMMY = 1, 100000
        IF (KEY(1) .NE. LE ) GO TO 740
        IF (KEY(2) .EQ. LND) GO TO 900
 740    READ(IMPS, 1700) KEY(1), KEY(2)
 750 CONTINUE
     GO TO 900

*   -----------------------------------
*   Not enough core to solve the problem.
*   -----------------------------------
 800 CALL M1PAGE(2)
     WRITE(IPRINT, 1120) MINCOR
     WRITE(ISUMM , 1120) MINCOR
     IERR   = 42

*    Exit.

 900 REWIND ISCR
     IF (IMPS .NE. IREAD  .AND.  IMPS .NE. ISPECS) REWIND IMPS
     RETURN

 1100 FORMAT(' EXIT -- Fatal errors in the MPS file')
 1110 FORMAT(' EXIT -- Not enough storage to read the MPS file')
 1120 FORMAT(' EXIT -- Not enough storage to start solving',
    $        ' the problem'
    $     // ' WORKSPACE (TOTAL)  should be significantly',
    $        ' more than', I8)
```

```
1300 FORMAT(/ ' Length of row-name hash table  ', I12
   $      / ' Collisions during table lookup ', I12)
1320 FORMAT(/ ' No. of rejected coefficients   ', I12)
1350 FORMAT(/ ' No. of Jacobian entries specified', I10)
1400 FORMAT(/ ' No. of INITIAL  BOUNDS  specified', I10
   $      / ' No. of superbasics specified   ', I12)
1500 FORMAT(/ ' PARTIAL PRICE section size (A) ', I12
   $      / ' PARTIAL PRICE section size (I) ', I12)
1700 FORMAT(A1, A2)

*     End of M3INPT
      END

*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

      SUBROUTINE M3MPSA( MROWS, MCOLS, MELMS, LENH, NCOLL, M,
     $                   NN, NNCON, KEY, NCARD,
     $                   HRTYPE, NAME1R, NAME2R, KEYNAM )

      IMPLICIT           REAL*8(A-H,O-Z)
      CHARACTER*4        KEY
      DIMENSION          KEY(3), NCARD(6)
      INTEGER*2          HRTYPE(MROWS)
      INTEGER            KEYNAM(LENH)
      INTEGER            NAME1R(MROWS), NAME2R(MROWS)

*     ------------------------------------------------------------------
*     M3MPSA  inputs the NAME and ROWS sections of an MPS file.
*
*     Original version written by Keith Morris, Wellington, 1973.
*     Modified 1975 to use a hash table for the row names.
*     Modified 1979 to treat the (NNCON by NNJAC) principal submatrix
*        as a Jacobian for nonlinear constraints.
*     Modified 1980 to add phantom columns to the end of  A.
*     Modified 1982 to store the RHS as bounds on the logicals,
*        instead of the last column of  A.  The constraints now have the
*        form    A*X  +  I*S  =  0,    BL .LE. (X, S) .LE. BU,    where
*        A  has  M  rows,  N  columns  and  NE  nonzero elements.
*     Modified 1982 to treat * in column 1 correctly, and to retry if
*        the MPS file is on disk and MROWS, MCOLS or MELMS are too small
*     Apr 1984: Added check for duplicate row entries in columns.
*     Mar 1985: Changes made to handle characters as in Fortran 77.
*     Oct 1985: M3MPS split into M3MPSA, M3MPSB, M3MPSC.
*               Revisions made to handle characters more efficiently.
*     ------------------------------------------------------------------

      COMMON    /M1FILE/ IREAD,IPRINT,ISUMM
      COMMON    /M2FILE/ IBACK,IDUMP,ILOAD,IMPS,INEWB,INSRT,
     $                   IOLDB,IPNCH,IPROB,ISCR,ISOLN,ISPECS,IREPRT
      COMMON    /M3MPS3/ AIJTOL,BSTRUC(2),MLST,MER,
     $                   AIJMIN,AIJMAX,NA0,LINE,IER(20)
      COMMON    /M3MPS4/ NAME(2),MOBJ(2),MRHS(2),MRNG(2),MBND(2),MINMAX
      COMMON    /M3MPS5/ AELEM(2), ID(6), IBLANK
      COMMON    /M5LOBJ/ SINF,WTOBJ,MINIMZ,NINF,IOBJ,JOBJ,KOBJ
      COMMON    /M5LOG1/ IDEBUG,IERR,LPRINT
      COMMON    /DUFILE/ IDUAL

      LOGICAL            FOUND, GOTNM
```

```
      CHARACTER*4      KEY2
      CHARACTER*4      LC,LN,LR
      CHARACTER*4      LAM,LOL,LOW
      CHARACTER*4      LEX,LGX,LLX,LNX,LXE,LXG,LXL,LXN
      CHARACTER*4      LBLANK,LMIN,LMAX
      DATA         LC,LN,LR        /'C','N','R'/
      DATA         LAM,LOL,LOW     /'AM','OL','OW'/
      DATA         LEX,LGX,LLX,LNX  /'E ','G ','L ','N '/
      DATA         LXE,LXG,LXL,LXN  /' E',' G',' L',' N'/
      DATA         LBLANK,LMIN,LMAX /'    ','MIN ','MAX '/
*     -----------------------------------------------------------

      WRITE(IPRINT, 1000)
      CALL M1CHAR( LBLANK, IBLANK )
      M      = 0
      IOBJ   = 0
      LINE   = 0
      GOTNM  = MOBJ(1) .NE. IBLANK
      INFORM = 0

      DO 2 I = 1, 20
        IER(I) = 0
    2 CONTINUE
      DO 6 I = 1, LENH
        KEYNAM(I) = 0
    6 CONTINUE

*     Look for the NAME card.

   10 CALL M3READ( 1, IMPS, LINE, 5, KEY, INFORM )
      IF (KEY(1) .EQ. LN  .AND.  KEY(2) .EQ. LAM) GO TO 15
      IF (IER(1) .GT. 0) GO TO 10
      IER(1) = 1
      WRITE(IPRINT, 1100)
      WRITE(ISUMM , 1100)
      GO TO 10

   15 NAME(1) = ID(3)
      NAME(2) = ID(4)
      WRITE(ISUMM, 5000) 'DUAL',NAME(1)
      WRITE(IDUAL, '(A4,11X,2A4)') 'NAME', 'DUAL', NAME(1)
      WRITE(ISCR, 2000) 'DUAL',NAME(1)
      WRITE(IDUAL, '(A4)') 'ROWS'
*     (dual) exchange  max <--> min
      MINIMZ = -MINIMZ
      IF (MINIMZ .GT. 0) CALL M1CHAR(LMIN,MINMAX)
      IF (MINIMZ .LE. 0) CALL M1CHAR(LMAX,MINMAX)

*     Look for the ROWS card.

      CALL M3READ( 1, IMPS, LINE, 5, KEY, INFORM )
      INFORM = 0
      IF (KEY(1) .EQ. LR  .AND.  KEY(2) .EQ. LOW) GO TO 30
      IER(1) = IER(1) + 1
      WRITE(IPRINT, 1120)
      WRITE(ISUMM , 1120)
      GO TO 35
```

```
*  ================================================================
*  Read the row names and check if the relationals are valid.
*  ================================================================
   30 CALL M3READ( 1, IMPS, LINE, MLST, KEY, INFORM )
      IF (INFORM .NE. 0) GO TO 110
   35 KEY2   = KEY(2)
      IF (KEY2 .EQ. LGX  .OR.  KEY2 .EQ. LXG) GO TO 40
      IF (KEY2 .EQ. LEX  .OR.  KEY2 .EQ. LXE) GO TO 50
      IF (KEY2 .EQ. LLX  .OR.  KEY2 .EQ. LXL) GO TO 60
      IF (KEY2 .EQ. LNX  .OR.  KEY2 .EQ. LXN) GO TO 70
      IER(3) = IER(3)+1
      IF (IER(3) .GT. MER) GO TO 30
      WRITE(IPRINT, 1160) LINE, (KEY(I), I=1,3), ID(1),ID(2)
      WRITE(ISUMM , 1160) LINE, (KEY(I), I=1,3), ID(1),ID(2)
      GO TO 30

*     Come here to process valid relational

   40 IT     = -1
      GO TO 80
   50 IT     = 0
      GO TO 80
   60 IT     = 1
      GO TO 80
   70 IT     = 2
      IF (IOBJ.GT.0) GO TO 80
      IF (  GOTNM  ) GO TO 75
      MOBJ(1) = ID(1)
      MOBJ(2) = ID(2)
*     (dual) the objective is the RHS of the dual.
      MRHS(1) = ID(1)
      MRHS(2) = ID(2)
      IF (NN .EQ. 0) GO TO 75
      WRITE(IPRINT, 1170) MOBJ(1),MOBJ(2)
      WRITE(ISUMM , 1170) MOBJ(1),MOBJ(2)

   75 IF (ID(1).NE.MOBJ(1) .OR. ID(2).NE.MOBJ(2)) GO TO 80
      IOBJ    = M + 1
      NCARD(1) = NCARD(1) + 1
      IF (IOBJ .LE. NNCON) GO TO 90

*        ..............................................................
*     Look up the row name  ID(1), ID(2)  in the hash table.
*        ..............................................................
   80 CALL M1HASH( LENH,MROWS,NCOLL,
     $            ID(1),ID(2),2,KEYNAM,NAME1R,NAME2R,IA,FOUND )
      IF (FOUND) GO TO 140

*     Enter new row name in hash table.

      M         = M + 1
      IF (M .GT. MROWS) GO TO 30
      KEYNAM(IA) = M
      NAME1R(M)  = ID(1)
      NAME2R(M)  = ID(2)
      HRTYPE(M)  = IT
      GO TO 30
```

```
*   Linear obj row is ahead of nonlinear rows -- fatal error.

 90 WRITE(IPRINT, 1180) MOBJ(1),MOBJ(2)
    WRITE(ISUMM , 1180) MOBJ(1),MOBJ(2)
    IERR   = 40
    RETURN

*   ============================================================
*   Should be COLUMNS card.  Error if no rows.
*   ============================================================
 110 IF (KEY(1) .EQ. LC  .AND.  KEY(2) .EQ. LOL) GO TO 115
    IER(1) = IER(1) + 1
    WRITE(IPRINT, 1130)
    WRITE(ISUMM , 1130)

*   (dual) Writting of row names out to the scratch file
*   is moved to DUBOUNDS.

 115 IF (M .LE.   0  ) GO TO 150
    IF (M .GT. MROWS) GO TO 160
    WRITE(ISUMM, 5100) M
    IF (IOBJ .EQ. 0 ) GO TO 155
    RETURN

*   Duplicate row names comes here.

 140 IER(4) = IER(4) + 1
    IF (IER(4).GT.MER) GO TO 30
    WRITE(IPRINT, 1200) ID(1),ID(2)
    WRITE(ISUMM , 1200) ID(1),ID(2)
    GO TO 30

*   No rows got past the sieve, so bomb off.

 150 WRITE(IPRINT, 1300)
    WRITE(ISUMM , 1300)
    IER(1) = IER(1) + 1
    IERR   = 40
    RETURN

*   No objective function.

 155 WRITE(IPRINT, 1600)
    WRITE(ISUMM , 1600)
    RETURN

*   Too many rows.

 160 WRITE(IPRINT, 3030) MROWS,M
    WRITE(ISUMM , 3030) MROWS,M
    IER(1) = IER(1) + 1
    IERR   = 41
    RETURN

 1000 FORMAT( 1' / ' MPS file' / ' --------')
 1100 FORMAT(' XXXX  Garbage before NAME card')
 1120 FORMAT(' XXXX  ROWS card not found')
 1130 FORMAT(' XXXX  COLUMNS card not found')
```

```
1160 FORMAT(' XXXX  Illegal row type at line', I7, '... ',A1,A2,A1,2A4)
1170 FORMAT(' XXXX  Note --- row  ', 2A4,
   $  ' selected as linear part of objective.')
1180 FORMAT(/ ' XXXX  The linear objective card      N ', 2A4
   $        / ' XXXX  is out of place.   Nonlinear constraints'
   $        / ' XXXX  must be listed first in the ROWS section.')
1200 FORMAT(' XXXX  Duplicate row name --', 2A4, ' -- ignored')
1300 FORMAT(' XXXX  No rows specified')
1600 FORMAT(' XXXX  Warning - no linear objective selected')
2000 FORMAT(2A4)
3030 FORMAT(' XXXX  Too many rows. Limit was', I8,
   $  4X, ' Actual number is', I8)
5000 FORMAT(' NAME    ', 2A4)
5100 FORMAT(' ROWS IN THE PRIMAL    ', I8)

*     End of M3MPSA
      END

*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

      SUBROUTINE M3MPSB( Z,NWCORE,IEXPLACE,MCOLS, MELMS, LENH, NCOLL,
     $                   M, N, NB, NE, NKA,
     $                   NN, NNCON, NNJAC, NNOBJ, NJAC, KEY, NCARD,
     $                   HRTYPE, NAME1R, NAME2R, KEYNAM,
     $                   KA, HA, A, BL, BU, KB, NAME1C, NAME2C)

      IMPLICIT          REAL*8(A-H,O-Z)
      CHARACTER*4       KEY
      DIMENSION         KEY(3), NCARD(6)
      INTEGER*2         HRTYPE(M), HA(MELMS)
      INTEGER           KA(NKA),   KB(M), KEYNAM(LENH)
      INTEGER           NAME1R(M),    NAME2R(M)
      INTEGER           NAME1C(MCOLS), NAME2C(MCOLS)
      DOUBLE PRECISION  A(MELMS), BL(NB), BU(NB), Z(NWCORE)

*     ---------------------------------------------------------------
*     M3MPSB inputs the COLUMNS, RHS and RANGES sections of an MPS file,
*     Transposes A and exchange tha names of the rows and the columns.
*     ---------------------------------------------------------------

      COMMON    /M1EPS / EPS,EPS0,EPS1,EPS2,EPS3,EPS4,EPS5,PLINFY
      COMMON    /M1FILE/ IREAD,IPRINT,ISUMM
      COMMON    /M2FILE/ IBACK,IDUMP,ILOAD,IMPS,INEWB,INSRT,
     $                   IOLDB,IPNCH,IPROB,ISCR,ISOLN,ISPECS,IREPRT
      COMMON    /M3MPS3/ AIJTOL,BSTRUC(2),MLST,MER,
     $                   AIJMIN,AIJMAX,NA0,LINE,IER(20)
      COMMON    /M3MPS4/ NAME(2),MOBJ(2),MRHS(2),MRNG(2),MBND(2),MINMAX
      COMMON    /M3MPS5/ AELEM(2), ID(6), IBLANK
      COMMON    /M5LOBJ/ SINF,WTOBJ,MINIMZ,NINF,IOBJ,JOBJ,KOBJ
      COMMON    /M5LOG1/ IDEBUG,IERR,LPRINT
      COMMON    /M8AL1 / PENPAR,ROWTOL,NCOM,NDEN,NLAG,NMAJOR,NMINOR
      COMMON    /CYCLCM/ CNVTOL,JNEW,MATERR,MAXCY,NEPHNT,NPHANT,NPRINT
      COMMON    /DUFILE/ IDUAL

      PARAMETER         ( ZERO = 0.0 )

      LOGICAL           FOUND, GOTNM
      CHARACTER*4       LR
```

```
      CHARACTER*4       LAN,LHS
      DATA              LR      /'R'/
      DATA              LAN,LHS /'AN','HS'/
*     ------------------------------------------------

      BPLUS  = PLINFY
      BMINUS = -BPLUS
      NMCOL1 = 1234
      NMCOL2 = 5678
      N      = 0
      NA0    = 0
      NE     = 0
      NE1    = -1
      NJAC   = 0
      INFORM = 0

*     (dual) Insert zeros to the first column of A for the
*     objective of the dual.

      DO 190 I = 1, M
       A(I) = ZERO
       HA(I) = I
190   CONTINUE
      NE = M
      KA(1) = 1
      N = 1
      NAME1C(1) = 'DUAL'
      NAME2C(1) = 'OBJC'

      DO 205 I = 1, M
        KB(I) = 0
 205  CONTINUE

*     ==========================================================
*     Read the next COLUMNS card.
*     ==========================================================
 210  CALL M3READ( 2, IMPS, LINE, MLST, KEY, INFORM )
      IF (INFORM .NE. 0) GO TO 310
      IF (ID(1) .NE. NMCOL1  .OR.  ID(2) .NE. NMCOL2) GO TO 300

*     Process two row names and values.

 220  DO 260 I  = 1, 2

*        Check for only one on the card.

         K     = I + I
         ID1   = ID(K+1)
         ID2   = ID(K+2)
         IF (ID1 .NE. IBLANK) GO TO 230
         IF (ID2 .EQ. IBLANK) GO TO 260

*        Look up the row name.

 230     CALL M1HASH( LENH, M, NCOLL,
     $               ID1, ID2, 1, KEYNAM, NAME1R, NAME2R, IA, FOUND )
         IF (FOUND) GO TO 240
         IER(5) = IER(5)+1
```

```
      IF (IER(5).LE.MER) WRITE(IPRINT, 1400) ID1,ID2,LINE
      GO TO 260

  240 AIJ   = AELEM(I)
      IROW  = KEYNAM(IA)

*     Test for a duplicate entry.

      IF (KB(IROW) .NE. N) GO TO 242
      IER(8) = IER(8) + 1
      IF (IER(8).LE.MER) WRITE(IPRINT, 1420) NMCOL1,NMCOL2,
     $                                 ID1,ID2,AIJ,LINE
      GO TO 260

  242 IF (IROW .LE. NNCON  .AND.  N .LE. NNJAC) GO TO 250
      IF (DABS(AIJ) .GE. AIJTOL) GO TO 255

*     Ignore small AIJs.

      NA0   = NA0 + 1
      GO TO 260

*     This is a Jacobian element. In the sparse case, make sure
*     it is squeezed in ahead of any linear-constraint elements.

  250 NJAC    = NJAC + 1
      IF (NDEN .EQ. 1) A(NE1 + IROW) = AIJ
      IF (NDEN .EQ. 1) GO TO 260
      LJAC    = LJAC + 1
      IF (LJAC .GT. NE) GO TO 255
      AIJ     = A(LJAC)
      IROW    = HA(LJAC)
      A(LJAC) = AELEM(I)
      HA(LJAC) = KEYNAM(IA)

*     Mark this row as having an entry, and pack the nonzero.

  255 KB(IROW) = N
      NE      = NE + 1
      IF (NE .GT. MELMS) GO TO 260
      HA(NE)  = IROW
      A(NE)   = AIJ
  260 CONTINUE
    GO TO 210

*   Come here for a new column.

  300 IF (NE.LE.NE1) GO TO 320
  301 N       = N + 1
    NE1       = NE
    NMCOL1    = ID(1)
    NMCOL2    = ID(2)
    IF (N .GT. MCOLS) GO TO 220
    KA(N)     = NE + 1
    NAME1C(N) = NMCOL1
    NAME2C(N) = NMCOL2

*   Make room for a Jacobian element.
```

```
      IF (NNCON .EQ. 0    ) GO TO 220
      LJAC   = NE
      IF (NDEN  .EQ. 2  .OR.  N .GT. NNJAC) GO TO 220
      NE     = NE + NNCON
      IF (NE    .GT. MELMS) GO TO 220
      NE     = NE - NNCON
      DO 302 I  = 1, NNCON
        NE     = NE + 1
        HA(NE) = I
        A(NE)  = ZERO
  302 CONTINUE
      GO TO 220

*     ===============================================================
*     See if we have hit the RHS.
*     ===============================================================
  310 IF (N  .GT. MCOLS) GO TO 360
      IF (NE .GT. MELMS) GO TO 370
      IF (NE .LE. NE1) GO TO 320
  311 IF (KEY(1) .EQ. LR  .AND.  KEY(2) .EQ. LHS) GO TO 340

*     NOPE SUMPINS RONG

      IER(7) = IER(7)+1
      WRITE(IPRINT, 1140)
      WRITE(ISUMM , 1140)
      GO TO 340

*     Column with no rows -- accept only if variable is nonlinear.
*     (Insert dummy column with zero in first row.)

  320 IF (N.LE.NN) GO TO 325
      IER(6) = IER(6)+1
      IF (IER(6).LE.MER) WRITE(IPRINT, 1500) NMCOL1,NMCOL2
  325 NE     = NE + 1
      HA(NE) = 1
      A(NE)  = ZERO
      IF (INFORM .EQ. 0) GO TO 301
      GO TO 311

*     Are there any columns at all?

  340 IF (N .GT. 0) GO TO 400
      WRITE(IPRINT, 1610)
      WRITE(ISUMM , 1610)
      IER(2) = IER(2) + 1
      IERR   = 40
      RETURN

*     Too many columns.

  360 N      = N + NPHANT
      WRITE(IPRINT, 3040) MCOLS,N
      WRITE(ISUMM , 3040) MCOLS,N
      GO TO 380

*     Too many elements.
```

```
 370 NE     = NE + NEPHNT
    WRITE(IPRINT, 3050) MELMS,NE
    WRITE(ISUMM , 3050) MELMS,NE
 380 IER(2) = IER(2) + 1
    IERR   = 41
    RETURN

 400 WRITE(ISUMM, 5200) N-1,NE-M

*   (dual) Transposing A:
    KA(N+1) = NE+1
    CALL TRANSPOS(A,HA,KA,Z(IEXPLACE),NE,NKA)
*   (dual) Exchange M,N. Save old values in MOLD,NOLD.
    MOLD = M
    NOLD = N
    N = MOLD
    M = NOLD

*   (dual) The treatment of the phantom elements and the
*    slacks was moved to subroutine DUBOUND.

*   (dual) Remove old IOBJ line (now column) from A(T) to Z(LPI+1)
    DO 4002 I = IEXPLACE,IEXPLACE+M
     Z(I) = ZERO
4002 CONTINUE
    K1 = KA(IOBJ)
    K2 = KA(IOBJ+1)
    IF (K1 .GE. K2) GO TO 4010
    DO 4005 I=K1,K2-1
     IR = IEXPLACE+HA(I)
     Z(IR) = A(I)
4005 CONTINUE
    DO 4008 I=K1,NE
     IF (I .GT. NE-K2+K1) GO TO 4006
     A(I) = A(K2+I-K1)
     HA(I) = HA(K2+I-K1)
     GO TO 4008
4006  A(I) = ZERO
     HA(I) = 0
4008 CONTINUE
    DO 4009 I=IOBJ,N+1
     KA(I) = KA(I+1)-K2+K1
4009 CONTINUE
    N = N-1
    NE = NE-K2+K1
4010 IOBJOLD = IOBJ
    IOBJ = 1

*   (dual) Excahnge the names of the rows and the columns
    CALL EXNAMES(M,N,IOBJ,IOBJOLD,
    $          KEYNAM,LENH,NAME1R,NAME2R,HRTYPE,MOLD,
    $          NAME1C,NAME2C,NOLD,Z(IEXPLACE+M+MOLD+1),
    $          Z(IEXPLACE+M+3*NOLD+3))

*   (dual) Set bounds of pi1.

4061 LOLDTP = IEXPLACE+M+3*NOLD+3
```

```
      DO 407 I = 1, N
        IF (Z(LOLDTP+I-1) .LT. (-0.5)) GO TO 4070
        IF (Z(LOLDTP+I-1) .LT. 0.5) GO TO 4075
        IF (Z(LOLDTP+I-1) .LT. 1.5) GO TO 4080
 4075   BL(I) = BMINUS
        BU(I) = BPLUS
        GO TO 407
 4070   IF (MINIMZ .GT. 0) GO TO 4072
 4071   BL(I) = 0
        BU(I) = BPLUS
        GO TO 407
 4072   BL(I) = BMINUS
        BU(I) = 0
        GO TO 407
 4080   IF (MINIMZ .GT. 0) GO TO 4071
        GO TO 4072
  407 CONTINUE

*     ---------------------------------------------------------
*     Input the RHS.
*     ---------------------------------------------------------

      IF (KEY(1) .NE. LR ) GO TO 490
      IF (KEY(2) .NE. LHS) GO TO 490
      GOTNM  = .FALSE.
      INFORM = 0

*     ===========================================================
*     Read next RHS card and see if it is the one we want.
*     Use it as the objective of the dual.
*     ===========================================================
  410 CALL M3READ( 2, IMPS, LINE, MLST, KEY, INFORM )
      IF (INFORM .NE. 0) GO TO 490
      IF (   GOTNM    ) GO TO 420
      MOBJ(1) = ID(1)
      MOBJ(2) = ID(2)
      CALL M1HASH(LENH,M,N,ID(1),ID(2),2,KEYNAM,NAME1R,NAME2R,IA,FOUND)
      KEYNAM(IA) = IOBJ
      NAME1R(IOBJ) = ID(1)
      NAME2R(IOBJ) = ID(2)
      HRTYPE(IOBJ) = 2
      GOTNM  = .TRUE.
  420 IF (ID(1) .NE. MOBJ(1)  .OR.  ID(2) .NE. MOBJ(2)) GO TO 410

*     Look at both halves of the record.

      DO 460 I  = 1, 2
        K     = I + I
        ID1   = ID(K+1)
        ID2   = ID(K+2)
        IF (ID1.EQ.IBLANK .AND. ID2.EQ.IBLANK) GO TO 460
        CALL DUALIST( N,ID1,ID2,NAME1C,NAME2C,IA,FOUND )
        IF (FOUND) GO TO 440
        IER(5) = IER(5)+1
        IF (IER(5).LE.MER) WRITE(IPRINT, 1400) ID1,ID2,LINE
        WRITE(IDUAL,'(A15,2A4)') ' COL NOT FOUND ',ID1,ID2
        GO TO 460
```

```
*      Another RHS element made it.
*      (dual) Use it as an element of the dual objective.

 440   BND    = AELEM(I)
      IF (DABS(BND) .LT. AIJTOL) GO TO 460
      NCARD(2) = NCARD(2)+1
      J = KA(IA)
 445   IF (J .GE. KA(IA+1)) GO TO 448
      IF (HA(J) .EQ. IOBJ) GO TO 450
      J = J+1
      GO TO 445
 448   WRITE(IDUAL,'(A8,A5,I5,2X,2A4)') ' ERROR11',' COL ',IA,
   $           NAME1C(IA),NAME2C(IA)
      GO TO 460
 450   A(J) = BND
 460 CONTINUE
    GO TO 410

*    RHS has been input.

 490 IF (NCARD(2) .GT. 0) GO TO 500
    WRITE(IPRINT, 1620)
    WRITE(ISUMM , 1620)

*    ----------------------------------------------------------
*    Input RANGES. (untreated for the dual case)
*    ----------------------------------------------------------

*    Check for no RANGES.

 500 IF (KEY(1) .NE. LR ) GO TO 590
    IF (KEY(2) .NE. LAN) GO TO 590
    GOTNM  = MRNG(1) .NE. IBLANK
    INFORM = 0

*    ==============================================================
*    Read card and see if it is the RANGE we want.
*    ==============================================================
 510 CALL M3READ( 2, IMPS, LINE, MLST, KEY, INFORM )
    IF (INFORM .NE. 0) GO TO 590
    IF (    GOTNM    ) GO TO 520
    MRNG(1) = ID(1)
    MRNG(2) = ID(2)
    GOTNM   = .TRUE.
 520 IF (ID(1).NE.MRNG(1) .OR. ID(2).NE.MRNG(2)) GO TO 510

*    Look at both halves of the record.

    DO 560 I  = 1, 2
      K     = I + I
      ID1   = ID(K+1)
      ID2   = ID(K+2)
      IF (ID1.EQ.IBLANK .AND. ID2.EQ.IBLANK) GO TO 560
      CALL M1HASH( LENH,M,NCOLL,
   $              ID1,ID2,1,KEYNAM,NAME1R,NAME2R,IA,FOUND )
      IF (FOUND) GO TO 550
      IER(5) = IER(5)+1
      IF (IER(5).LE.MER) WRITE(IPRINT, 1400) ID1,ID2,LINE
```

```
      GO TO 560

*     Another  RANGE element made it.

 550   BRNG   = AELEM(I)
      ARNG   = DABS(BRNG)
      NCARD(3) = NCARD(3)+1
      IROW   = KEYNAM(IA)
      JSLACK = N + IROW
      K      = HRTYPE(IROW)
      IF (K.LT.0) BL(JSLACK) = BU(JSLACK) - ARNG
      IF (K.GT.0) BU(JSLACK) = BL(JSLACK) + ARNG
      IF (K.EQ.0 .AND. BRNG.GT.ZERO) BL(JSLACK) = BU(JSLACK) - ARNG
      IF (K.EQ.0 .AND. BRNG.LT.ZERO) BU(JSLACK) = BL(JSLACK) + ARNG
 560 CONTINUE
     GO TO 510

*    End of RANGES.
 590 RETURN

1140 FORMAT(' XXXX  RHS card not found')
1400 FORMAT(' XXXX  Non-existent row    specified -- ', 2A4,
   $  ' -- entry ignored in line', I7)
1420 FORMAT(' XXXX  Column  ', 2A4, '  has more than one entry',
   $       ' in row  ', 2A4
   $     / ' XXXX     Coefficient', 1PE15.5, '  ignored in line', I10)
1500 FORMAT(' XXXX  No valid row entries in column  ', 2A4)
1610 FORMAT(' XXXX  No columns specified')
1620 FORMAT(' XXXX  Warning - the RHS is zero')
2000 FORMAT(2A4)
3040 FORMAT(' XXXX  Too many COLUMNS.   The limit was', I8,
   $  4X, ' Actual number is', I8)
3050 FORMAT(' XXXX  Too many ELEMENTS.  The limit was', I8,
   $  4X, ' Actual number is', I8)
5200 FORMAT(' COLUMNS IN THE PRIMAL', I8 /
   $          ' ELEMENTS IN THE PRIMAL', I7)

*    End of M3MPSB
     END

*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

     SUBROUTINE DUBOUNDS( M, N, NE, NKA, MELMS, KEY, NCARD,
    $                    MCOLS, NB, A, HA, KA, HRTYPE,
    $                    NAME1C, NAME2C, NAME1R, NAME2R,
    $                    BL, BU, VRHS )

     IMPLICIT          REAL*8(A-H,O-Z)
     CHARACTER*4          KEY
     DIMENSION          KEY(3), NCARD(6)
     INTEGER*2          HRTYPE(M), HA(MELMS)
     INTEGER          KA(NKA), NAME1R(M), NAME2R(M)
     INTEGER           NAME1C(MCOLS), NAME2C(MCOLS)
     DOUBLE PRECISION   A(MELMS), BL(NB), BU(NB), VRHS(M)

*    ------------------------------------------------------------
*    DUBOUNDS inputs the BOUNDS section of an MPS file, set the bounds
*    of the dual variables and the slacks and writes the dual MPS file
```

```
*    into IDUAL.
*    ----------------------------------------------------------

      COMMON    /M1EPS / EPS,EPS0,EPS1,EPS2,EPS3,EPS4,EPS5,PLINFY
      COMMON    /M1FILE/ IREAD,IPRINT,ISUMM
      COMMON    /M2FILE/ IBACK,IDUMP,ILOAD,IMPS,INEWB,INSRT,
     $                   IOLDB,IPNCH,IPROB,ISCR,ISOLN,ISPECS,IREPRT
      COMMON    /M3MPS3/ AIJTOL,BSTRUC(2),MLST,MER,
     $                   AIJMIN,AIJMAX,NA0,LINE,IER(20)
      COMMON    /M3MPS4/ NAME(2),MOBJ(2),MRHS(2),MRNG(2),MBND(2),MINMAX
      COMMON    /M3MPS5/ AELEM(2), ID(6), IBLANK
      COMMON    /M5LOBJ/ SINF,WTOBJ,MINIMZ,NINF,IOBJ,JOBJ,KOBJ
      COMMON    /M5LOG1/ IDEBUG,IERR,LPRINT
      COMMON    /M8AL1 / PENPAR,ROWTOL,NCOM,NDEN,NLAG,NMAJOR,NMINOR
      COMMON    /CYCLCM/ CNVTOL,JNEW,MATERR,MAXCY,NEPHNT,NPHANT,NPRINT
      COMMON    /DUFILE/ IDUAL

      PARAMETER      ( ZERO = 0.0 )

      LOGICAL        GOTNM, IGNORE
      CHARACTER*4       LB   , LOU, LE , LND
      CHARACTER*4       LFR  , LFX, LLO, LMI, LPL, LUP
      CHARACTER*4       LINIT, LIAL, LID1, LID2
      DATA           LB   , LOU, LE , LND /'B','OU','E','ND'/
      DATA           LFR  , LFX, LLO      /'FR','FX','LO'/
      DATA           LMI  , LPL, LUP      /'MI','PL','UP'/
      DATA           LINIT, LIAL          /'INIT', 'IAL '/
*    ----------------------------------------------------------

      CALL M1CHAR( LINIT, IINIT )
      CALL M1CHAR( LIAL , IIAL  )
      INFORM = 1
      BPLUS  = PLINFY
      BMINUS = -BPLUS
      NPIS = 0

*     (dual) set default rows type
      J = 1
      IF (MINIMZ .GT. 0) J = -1
      DO 100 I=1,M
        HRTYPE(I) = J
  100 CONTINUE
      HRTYPE(IOBJ) = 2

*     Check for no BOUNDS.

      IF (KEY(1) .NE. LB ) GO TO 900
      IF (KEY(2) .NE. LOU) GO TO 900
      GOTNM  = MBND(1). NE. IBLANK
      INFORM = 0
      JMARK  = 1

*    ==============================================================
*     Read and check BOUNDS cards.  Notice the double plural.
*    ==============================================================
  610 CALL M3READ( 3, IMPS, LINE, MLST, KEY, INFORM )
      IF (INFORM .NE. 0) GO TO 900
      BND    = AELEM(1)
```

```
      IF (ID(1) .EQ. IINIT   .AND.  ID(2) .EQ. IIAL) GO TO 898
      IF (    GOTNM    ) GO TO 620
      MBND(1) = ID(1)
      MBND(2) = ID(2)
      GOTNM   = .TRUE.
  620 IF (ID(1) .NE. MBND(1)  .OR.  ID(2) .NE. MBND(2)) GO TO 610

*     Find which column name.

      CALL M4NAME( M, NAME1R, NAME2R, ID(3), ID(4),
     $             LINE, IER(10), 0, 1, M, JMARK, J )
      IF (J .GT. 0) GO TO 630
      IF (IER(10) .LE. MER) WRITE(IPRINT, 1400) ID(3),ID(4),LINE
      GO TO 610

*     Select BOUND type for column J.

  630 NCARD(4) = NCARD(4) + 1
      IF (KEY(2) .EQ. LUP) GO TO 660
      IF (KEY(2) .EQ. LLO) GO TO 670
      IF (KEY(2) .EQ. LFX) GO TO 680
      IF (KEY(2) .EQ. LFR) GO TO 690
      IF (KEY(2) .EQ. LMI) GO TO 695
      IF (KEY(2) .EQ. LPL) GO TO 696

*     This lad didn't even make it to Form 1.

      IER(11) = IER(11) + 1
      IF (IER(11) .GT. MER) GO TO 610
      WRITE(IPRINT, 1700) LINE, (KEY(I), I=1,3), (ID(I), I=1,4)
      GO TO 610

*     Process each BOUND type.

*     (dual) Bounds treated as constraints.
*     (dual) x(j) < bnd
  660 K = KA(N+1)
      A(K) = BND
      HA(K) = IOBJ
      A(K+1) = 1
      HA(K+1) = J
      NE = NE+2
      N = N+1
      KA(N+1) = NE+1
      IF (MINIMZ .GT. 0) BL(N) = 0
      IF (MINIMZ .GT. 0) BU(N) = BPLUS
      IF (MINIMZ .LE. 0) BL(N) = BMINUS
      IF (MINIMZ .LE. 0) BU(N) = 0
      K = NPIS/10000
      IF (K .GT. 0) CALL DUIXTOS(K,LID1)
      IF (K .LE. 0) LID1 = 'DUAL'
      K = MOD(NPIS,10000)
      CALL DUIXTOS(K,LID2)
      CALL M1CHAR(LID1,ID1)
      CALL M1CHAR(LID2,ID2)
      NAME1C(N) = ID1
      NAME2C(N) = ID2
      NPIS = NPIS+1
```

```
      GO TO 610
*     (dual) x(j) > bnd
  670 K = KA(N+1)
      A(K) = BND
      HA(K) = IOBJ
      A(K+1) = 1
      HA(K+1) = J
      NE = NE+2
      N = N+1
      KA(N+1) = NE+1
      IF (MINIMZ .GT. 0) BL(N) = BMINUS
      IF (MINIMZ .GT. 0) BU(N) = 0
      IF (MINIMZ .LE. 0) BL(N) = 0
      IF (MINIMZ .LE. 0) BU(N) = BPLUS
      K = NPIS/10000
      IF (K .GT. 0) CALL DUIXTOS(K,LID1)
      IF (K .LE. 0) LID1 = 'DUAL'
      K = MOD(NPIS,10000)
      CALL DUIXTOS(K,LID2)
      CALL M1CHAR(LID1,ID1)
      CALL M1CHAR(LID2,ID2)
      NAME1C(N) = ID1
      NAME2C(N) = ID2
      NPIS = NPIS+1
      GO TO 610
*     (dual) x(j) = bnd
  680 K = KA(N+1)
      A(K) = BND
      HA(K) = IOBJ
      A(K+1) = 1
      HA(K+1) = J
      NE = NE+2
      N = N+1
      KA(N+1) = NE+1
      BU(N) = BPLUS
      BL(N) = BMINUS
      K = NPIS/10000
      IF (K .GT. 0) CALL DUIXTOS(K,LID1)
      IF (K .LE. 0) LID1 = 'DUAL'
      K = MOD(NPIS,10000)
      CALL DUIXTOS(K,LID2)
      CALL M1CHAR(LID1,ID1)
      CALL M1CHAR(LID2,ID2)
      NAME1C(N) = ID1
      NAME2C(N) = ID2
      NPIS = NPIS+1
      GO TO 610

*     (dual) Bounds that detrmine row types.
*     (dual) bminus < x(j) < bplus
  690 HRTYPE(J) = 0
      GO TO 610
*     (dual) x(j) < 0
  695 IF (MINIMZ .GT. 0) HRTYPE(J) = 1
      IF (MINIMZ .LE. 0) HRTYPE(J) = -1
      GO TO 610
*     (dual) x(j) > 0
  696 IF (MINIMZ .GT. 0) HRTYPE(J) = -1
```

```
      IF (MINIMZ .LE. 0) HRTYPE(J) = 1
      GO TO 610

  898 WRITE(IPRINT,1730)
      IF (NCARD(4) .GT. 0) GO TO 900
      MBND(1) = IBLANK
      MBND(2) = IBLANK
      WRITE(IPRINT, 1720)

*     (dual) Print the dual file and the scratch file.

  900 CH = 'N '
      DO 4100 I=1,M
        IF (HRTYPE(I) .LE. -1) CH = 'G '
        IF (HRTYPE(I) .EQ.  0) CH = 'E '
        IF (HRTYPE(I) .EQ.  1) CH = 'L '
        WRITE(IDUAL,1200) CH,NAME1R(I),NAME2R(I)
        WRITE(ISCR, 2000) NAME1R(I),NAME2R(I)
 4100 CONTINUE
      WRITE(IDUAL,'(A7)') 'COLUMNS'
      DO 4110 I=1,N
      DO 4120 J=KA(I),KA(I+1)-1
        IF (DABS(A(J)) .LT. AIJTOL) GO TO 4120
        K = HA(J)
        WRITE(IDUAL,'(4X,2A4,2X,2A4,2X,F12.4)') NAME1C(I),NAME2C(I),
     $          NAME1R(K),NAME2R(K),A(J)
 4120 CONTINUE
 4110 CONTINUE
      WRITE(ISCR, 2000) (NAME1C(I),NAME2C(I), I=1,N)
      WRITE(IDUAL,'(A3)') 'RHS'
      DO 4200 I=1,M
        IF (DABS(VRHS(I)) .LT. AIJTOL) GO TO 4200
        WRITE(IDUAL,1250) MRHS(1),MRHS(2),NAME1R(I),NAME2R(I),VRHS(I)
 4200 CONTINUE
      WRITE(IDUAL,'(A6)') 'BOUNDS'
      DO 4300 I=1,N
        CH = 'FR'
        IF (DABS(BL(I)) .LE. 1 .AND. BU(I) .GE. 100) CH = 'PL'
        IF (DABS(BU(I)) .LE. 1 .AND. BL(I) .LE. -100) CH = 'MI'
        WRITE(IDUAL,1300) CH,'DUBOUND ',NAME1C(I),NAME2C(I)
 4300 CONTINUE
      WRITE(IDUAL,'(A6)') 'ENDATA'

*     (dual) Insert Phantom Columns for Cycling algorithm.

      IF (NPHANT .LE. 0) GO TO 5006
      NEPHNT = MAX0(NEPHNT,NPHANT)
      IF (N+NPHANT .GT. MCOLS) GO TO 8060
      IF (NE+NEPHNT .GT. MELMS) GO TO 8070
      DO 5002 I=1,NEPHNT
        J = NE+I
        HA(J) = 1
        A(J) = ZERO
 5002 CONTINUE
      DO 5003 K=1,NPHANT
        N = N+1
        NE = NE+1
        WRITE(ISCR,5004) K
```

```
      NAME1C(N) = 0
      NAME2C(N) = 0
      KA(N) = NE
      IF (K .EQ. 1) NE = NE+NEPHNT-NPHANT
 5003 CONTINUE
    KA(N+1) = NE+1

*   (dual) Initialize bound for slacks.

 5006 DO 5008 I=1,M
      K = HRTYPE(I)
      JSLACK = N+1
      IF (K .LT. 0) BL(JSLACK) = BMINUS
      IF (K .LE. 0) BU(JSLACK) = ZERO
      IF (K .GE. 0) BL(JSLACK) = ZERO
      IF (K .GT. 0) BU(JSLACK) = BPLUS
      IF (K .EQ. 2) BL(JSLACK) = BMINUS
      BND = VRHS(I)
      IF (DABS(BND) .LT. AIJTOL) GO TO 5008
       IF (K .EQ. 2) GO TO 5008
      IF (K .LE. 0) BU(JSLACK) = -BND
      IF (K .GE. 0) BL(JSLACK) = -BND
 5008 CONTINUE

      RETURN

 8060 N = N+NPHANT
      WRITE(IPRINT,8040) MCOLS,N
      WRITE(ISUMM, 8040) MCOLS,N
      GO TO 8080
 8070 NE = NE+NEPHNT
      WRITE(IPRINT,8050) MELMS,NE
      WRITE(ISUMM, 8050) MELMS,NE
 8080 IER(2) = IER(2)+1
      IERR = 41
      RETURN

 1200 FORMAT(1X,A2,1X,2A4)
 1250 FORMAT(4X,2A4,2X,2A4,2X,F12.4)
 1300 FORMAT(1X,A2,1X,A8,2X,2A4)
 1400 FORMAT(' XXXX  Non-existent column specified -- ', 2A4,
     $  ' -- entry ignored in line', I7)
 1700 FORMAT(' XXXX  Illegal BOUND type at line', I7, '... ',
     $  A1,A2,A1, 2A4,2X,2A4)
 1720 FORMAT(' XXXX  Warning - first BOUNDS set is  INITIAL .',
     $  '  Other bounds will be ignored.')
 1730 FORMAT(' XXXX  Warning - INITIAL bounds where given to ',
     $  ' the primal variables. ')
 2000 FORMAT(2A4)
 5004 FORMAT('PHNT',I4)
 8040 FORMAT(' XXXX  Too many COLUMNS.   The limit was', I8,
     $  4X, ' Actual number is', I8)
 8050 FORMAT(' XXXX  Too many ELEMENTS.  The limit was', I8,
     $  4X, ' Actual number is', I8)

*     End of DUBOUNDS
      END
```

```
*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

      SUBROUTINE M3MPSC( M, N, NB, NE, NS,
     $                   KEY, NCARD,
     $                   BL, BU, HS, XN, NAME1C, NAME2C )

      IMPLICIT           REAL*8(A-H,O-Z)
      CHARACTER*4        KEY
      DIMENSION          KEY(3), NCARD(6)
      INTEGER*2          HS(NB)
      INTEGER            NAME1C(N), NAME2C(N)
      DOUBLE PRECISION   BL(NB), BU(NB), XN(NB)

      COMMON    /M1EPS / EPS,EPS0,EPS1,EPS2,EPS3,EPS4,EPS5,PLINFY
      COMMON    /M1FILE/ IREAD,IPRINT,ISUMM
      COMMON    /M2FILE/ IBACK,IDUMP,ILOAD,IMPS,INEWB,INSRT,
     $                   IOLDB,IPNCH,IPROB,ISCR,ISOLN,ISPECS,IREPRT
      COMMON    /M3MPS3/ AIJTOL,BSTRUC(2),MLST,MER,
     $                   AIJMIN,AIJMAX,NA0,LINE,IER(20)
      COMMON    /M3MPS4/ NAME(2),MOBJ(2),MRHS(2),MRNG(2),MBND(2),MINMAX
      COMMON    /M3MPS5/ AELEM(2), ID(6), IBLANK
      COMMON    /M5LOG1/ IDEBUG,IERR,LPRINT
      COMMON    /CYCLCM/ CNVTOL,JNEW,MATERR,MAXCY,NEPHNT,NPHANT,NPRINT

      PARAMETER          ( ZERO = 0.0 )

      LOGICAL            GOTNM, IGNORE
      CHARACTER*4        LB  , LOU, LE , LND
      CHARACTER*4        LFR , LFX, LLO, LMI, LPL, LUP
      CHARACTER*4        LINIT, LIAL
      DATA               LB  , LOU, LE , LND /'B','OU','E','ND'/
      DATA               LFR , LFX, LLO      /'FR','FX','LO'/
      DATA               LMI , LPL, LUP      /'MI','PL','UP'/
      DATA               LINIT, LIAL         /'INIT', 'IAL '/
*     ------------------------------------------------------------------

      CALL M1CHAR( LINIT, IINIT )
      CALL M1CHAR( LIAL , IIAL  )
      INFORM = 1
      BPLUS  = PLINFY
      BMINUS = -BPLUS

*     Fix phantom variables at zero.

      J1     = N - NPHANT + 1
      DO 604 J = J1, N
        BL(J) = ZERO
        BU(J) = ZERO
  604 CONTINUE

*     INITIAL BOUNDS set found.

*     ------------------------------------------------------------------
*     End of normal BOUNDS -- process the INITIAL BOUNDS set.
*     ------------------------------------------------------------------
  700 NS     = 0
      BPLUS  = 0.9*BPLUS
      BMINUS = -BPLUS
```

```
*    Set variables to be nonbasic at zero (as long as that's feasible).

     DO 706 J = 1, NB
       XN(J) = DMAX1 ( ZERO , BL(J) )
       XN(J) = DMIN1 ( XN(J), BU(J) )
       HS(J) = 0
       IF (XN(J) .EQ. BU(J)) HS(J) = 1
 706 CONTINUE

*    Ignore INITIAL BOUNDS if a basis will be loaded.

     IF (INFORM .NE. 0) GO TO 790
     IGNORE = IOLDB.GT.0 .OR. INSRT.GT.0 .OR. ILOAD.GT.0
     IF (  IGNORE    ) GO TO 710
     JMARK  = 1
     GO TO 720

*    ==============================================================
*    Read INITIAL BOUNDS set.
*    ==============================================================
 710 CALL M3READ( 3, IMPS, LINE, MLST, KEY, INFORM )
     IF (INFORM .NE. 0) GO TO 790
     BND    = AELEM(1)
     IF (IGNORE .OR. ID(1).NE.IINIT .OR. ID(2).NE.IIAL) GO TO 710

*    Find which column.

 720 CALL M4NAME( N, NAME1C, NAME2C, ID(3), ID(4),
    $             LINE, IER(12), 0, 1, N, JMARK, J )
     IF (J .GT. 0) GO TO 730
     IF (IER(12) .LE. MER) WRITE(IPRINT, 1400) ID(3),ID(4),LINE
     GO TO 710

*    Select BOUND type for column J.

 730 NCARD(6) = NCARD(6)+1
     JS       = 6
     IF (KEY(2) .EQ. LFR) JS = -1
     IF (KEY(2) .EQ. LFX) JS =  2
     IF (KEY(2) .EQ. LLO) JS =  0
     IF (KEY(2) .EQ. LUP) JS =  1
     IF (KEY(2) .EQ. LMI) JS =  4
     IF (KEY(2) .EQ. LPL) JS =  5
     IF (JS     .NE. 6  ) GO TO 750
     IER(13) = IER(13) + 1
     IF (IER(13) .GT. MER) GO TO 710
     WRITE(IPRINT, 1700) LINE, (KEY(I), I=1,3), (ID(I), I=1,4)
     GO TO 710

*    Process each type.

 750 IF (JS .EQ. 2) NS  = NS + 1
     IF (JS .EQ. 0) BND = BL(J)
     IF (JS .EQ. 0) JS  = 4
     IF (JS .EQ. 1) BND = BU(J)
     IF (JS .EQ. 1) JS  = 5
     IF (DABS(BND) .GE. BPLUS) BND = ZERO
```

```
      XN(J)  = BND
      HS(J)  = JS
      GO TO 710

*     ENDATA card.

  790 IF (KEY(1) .EQ. LE  .AND.  KEY(2) .EQ. LND) GO TO 800
      IER(14) = 1
      WRITE(IPRINT, 1150)
      WRITE(ISUMM , 1150)

*     ------------------------------------------------------------
*     Pass the buck - not got to Truman yet.
*     ------------------------------------------------------------

*     Check that  BL .LE. BU

  800 DO 802 J = 1, N
         B1    = BL(J)
         B2    = BU(J)
         IF (B1 .LE. B2) GO TO 802
         IER(20) = IER(20) + 1
         IF (IER(20) .LE. MER) WRITE(IPRINT, 1740) J, B1, B2
         BL(J) = B2
         BU(J) = B1
  802 CONTINUE

*     Count the errors.

      K      = 0
      DO 804 I = 1, 20
         K      = K + IER(I)
  804 CONTINUE
      IF (K .GT. 0) WRITE(IPRINT, 1900) K
      IF (K .GT. 0) WRITE(ISUMM , 1900) K
      WRITE(IPRINT, 2100) MOBJ(1),MOBJ(2),MINMAX,NCARD(1),
     $                    MRHS(1),MRHS(2),NCARD(2),
     $                    MRNG(1),MRNG(2),NCARD(3),
     $                    MBND(1),MBND(2),NCARD(4)
      RETURN

 1150 FORMAT(' XXXX  ENDATA card not found')
 1400 FORMAT(' XXXX  Non-existent column specified -- ', 2A4,
     $   ' -- entry ignored in line', I7)
 1700 FORMAT(' XXXX  Illegal BOUND type at line', I7, '... ',
     $   A1,A2,A1, 2A4,2X,2A4)
 1740 FORMAT(/' XXXX  Bounds back to front on column', I6,' :',
     $   1PE15.5, E15.5)
 1900 FORMAT(/' XXXX  Total no. of errors in MPS file', I6)
 2100 FORMAT(///
     $   ' Names selected' /
     $   ' --------------' /
     $   ' OBJECTIVE', 6X, 2A4, ' (', A3, ')', I8 /
     $   ' RHS      ', 6X, 2A4, I14 /
     $   ' RANGES   ', 6X, 2A4, I14 /
     $   ' BOUNDS   ', 6X, 2A4, I14)

*     End of M3MPSC
```

```
      END
*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

      SUBROUTINE M3READ( MODE, IMPS, LINE, MXLIST, KEY, INFORM )

      IMPLICIT         REAL*8(A-H,O-Z)
      CHARACTER*4      KEY
      DIMENSION        KEY(3)

*     ------------------------------------------------------------
*     M3READ  reads data from file IMPS and prints a listing on file
*     IPRINT. The data is assumed to be in MPS format, with items of
*     interest in the following six fields...
*
*     Field:    1       2       3       4       5       6
*
*     Columns: 01-04   05-12   15-22   25-36   40-47   50-61
*
*     Format:A1,A2,A1   2A4     2A4    E12.0    2A4    E12.0
*
*     Data: KEY(1-3)  ID(1-2)  ID(3-4)  AELEM(1)  ID(5-6)  AELEM(2)
*
*
*     Comment cards may contain a * in column 1 and anything in
*     columns 2-22. They are listed and then ignored.
*
*
*     MODE    specifies which fields are to be processed.
*
*     INFORM  is set to 1 if column 1 is not blank or *.
*
*     This version of M3READ written October 4, 1985.
*
*     IBUF bit added by J. Stone 5/89 so as to work with NDP compiler.
*       It should also work with anything else.
*     ------------------------------------------------------------

      COMMON     /M1FILE/ IREAD,IPRINT,ISUMM
      COMMON     /M3MPS5/ AELEM(2), ID(6), IBLANK
      CHARACTER*4       LBLANK     , LSTAR
      CHARACTER*64      IBUF
      DATA              LBLANK/'  '/, LSTAR/'*'/

   10 READ(IMPS, '(A)') IBUF
      GO TO (100, 200, 300), MODE

*     ------------------------------
*     MODE = 1  (NAME, ROWS)
*     ------------------------------
  100 READ(IBUF, 1000) KEY(1), KEY(2), KEY(3),
     $               ID(1), ID(2), ID(3), ID(4)
      LINE   = LINE + 1
      IF (KEY(1) .NE. LBLANK) GO TO 500
      IF (LINE   .LE. MXLIST)
     $WRITE(IPRINT, 2000) LINE, (KEY(I), I=1,3), (ID(I), I=1,4)
      RETURN
```

```
*     ------------------------------
*     MODE = 2  (COLUMNS, RHS, RANGES)
*     ------------------------------
  200 READ(IBUF, 1000) KEY(1), KEY(2), KEY(3), ID(1), ID(2),
     $              ID(3), ID(4), AELEM(1), ID(5), ID(6), AELEM(2)
      LINE   = LINE + 1
      IF (KEY(1) .NE. LBLANK) GO TO 500
      IF (LINE   .GT. MXLIST) RETURN
      IF (ID(5)  .EQ. IBLANK  .AND.  ID(6) .EQ. IBLANK) GO TO 310
      WRITE(IPRINT, 2000) LINE, (KEY(I), I=1,3), ID(1), ID(2),
     $              ID(3), ID(4), AELEM(1), ID(5), ID(6), AELEM(2)
      RETURN

*     ------------------------------
*     MODE = 3  (BOUNDS)
*     ------------------------------
  300 READ(IBUF, 1000) KEY(1), KEY(2), KEY(3), ID(1), ID(2),
     $              ID(3), ID(4), AELEM(1)
      LINE   = LINE + 1
      IF (KEY(1) .NE. LBLANK) GO TO 500
      IF (LINE   .GT. MXLIST) RETURN
  310 WRITE(IPRINT, 2000) LINE, (KEY(I), I=1,3), ID(1), ID(2),
     $              ID(3), ID(4), AELEM(1)
      RETURN

*     ------------------------------
*     Nonblank column 1.
*     ------------------------------
  500 WRITE(IPRINT, 2000) LINE, (KEY(I), I=1,3), (ID(I), I=1,4)
      IF (KEY(1) .EQ. LSTAR) GO TO 10
      INFORM = 1
      RETURN

 1000 FORMAT(      A1,A2,A1,2A4,2X,2A4,2X,:  E12.0,3X,2A4,2X,  E12.0)
 2000 FORMAT(I7,4X,A1,A2,A1,2A4,2X,2A4,2X,:1PE12.5,3X,2A4,2X,1PE12.5)

*     End of M3READ
      END

*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

      SUBROUTINE M2CORE( MODE, MINCOR )
      IMPLICIT           REAL*8(A-H,O-Z)

*     ------------------------------------------------------------
*     M2CORE  allocates all array storage for MINOS,
*     using various dimensions stored in COMMON blocks as follows:
*        (M2LEN )  MROWS, MCOLS, MELMS
*        (M3LEN )  NSCL
*        (M5LEN )  MAXR, MAXS, NN
*        (M7LEN )  NNOBJ
*        (M8LEN )  NJAC, NNCON, NNJAC
*
*     If MODE = 1, the call is from MINOS1 to estimate the storage
*                  needed for all stages of the problem.
*     IF MODE = 2, the call is from M3INPT to find the minimum storage
*                  needed for MPS input.
*     IF MODE = 3, all dimensions should be known exactly.
```

```
*           Normally the call is from M3INPT to find the minimum
*           storage to solve the problem and output the solution.
*           The basis factorization routines will say later
*           if they have sufficient storage.
*     ------------------------------------------------------------

      COMMON    /M1WORD/ NWORDR,NWORDI,NWORDH
      COMMON    /M2LEN / MROWS,MCOLS,MELMS
      COMMON    /M2MAPA/ NE    ,NKA   ,LA    ,LHA   ,LKA
      COMMON    /M2MAPZ/ MAXW  ,MAXZ
      COMMON    /M3LEN / M     ,N     ,NB    ,NSCL
      COMMON    /M3LOC / LASCAL,LBL   ,LBU   ,LBBL  ,LBBU  ,
     $                   LHRTYP,LHS   ,LKB
      COMMON    /M3MPS1/ LNAM1C,LNAM1R,LNAM2C,LNAM2R,LKEYNM
      COMMON    /M3MPS2/ LCNAM1,LRNAM1,LCNAM2,LRNAM2,LXS,LXL,LFREE
      COMMON    /M3SCAL/ SCLOBJ,SCLTOL,LSCALE
      COMMON    /M5LEN / MAXR  ,MAXS  ,MBS   ,NN    ,NN0   ,NR    ,NX
      COMMON    /M5LOC / LPI   ,LPI2  ,LW    ,LW2   ,
     $                   LX    ,LX2   ,LY    ,LY2   ,
     $                   LGSUB ,LGSUB2,LGRD  ,LGRD2 ,
     $                   LR    ,LRG   ,LRG2  ,LXN
      COMMON    /M7LEN / FOBJ  ,FOBJ2 ,NNOBJ ,NNOBJ0
      COMMON    /M7LOC / LGOBJ ,LGOBJ2
      COMMON    /M7CG2 / LCG1,LCG2,LCG3,LCG4,MODTCG,NITNCG,NSUBSP
      COMMON    /M8LEN / NJAC  ,NNCON ,NNCON0,NNJAC
      COMMON    /M8LOC / LFCON ,LFCON2,LFDIF ,LFDIF2,LFOLD ,
     $                   LBLSLK,LBUSLK,LXLAM ,LRHS  ,
     $                   LGCON ,LGCON2,LXDIF ,LXOLD
      COMMON    /M8AL1 / PENPAR,ROWTOL,NCOM,NDEN,NLAG,NMAJOR,NMINOR
*     ------------------------------------------------------------

      M      = MROWS
      N      = MCOLS
      NE     = MELMS
      MBS    = M + MAXS
      NKA    = N + 1
      NB     = N + M
      NSCL   = NB
      IF (LSCALE .EQ. 0) NSCL = 1

*     Allocate core for the constraint matrix  A.

      LA     = MAXW   + 1
      LHA    = LA     + NE
      LKA    = LHA    + 1 + (NE /NWORDH)
      MINZ   = LKA    + 1 + (NKA/NWORDI)

*     BL, BU, etc.

      NN0    = MAX0( NN   , 1 )
      NNCON0 = MAX0( NNCON, 1 )
      NNOBJ0 = MAX0( NNOBJ, 1 )
      MAXR   = MAX0( MAXR , 1 )
      NR     = MAXR*(MAXR + 1)/2  +  (MAXS - MAXR)
      NX     = MAX0( MBS, NN )
      NX2    = 0
      IF (LSCALE .EQ. 2) NX2 = NN
```

```
      LBL    = MINZ
      LBU    = LBL    + NB
      LKB    = LBU    + NB
      LASCAL = LKB    + 1 + (MBS  /NWORDI)
      LHRTYP = LASCAL + NSCL
      LHS    = LHRTYP + 1 + (MBS  /NWORDH)
      MINZ   = LHS    + 1 + (NB   /NWORDH)

*     LP and Reduced-gradient algorithm.
***** BEWARE -- length  0000  is used below for arrays that are
***** not needed in this version of MINOS.

      LPI    = MINZ
      LPI2   = LPI    + M
      LW     = LPI2   + 0000
      LW2    = LW     + NX
      LX     = LW2    + 0000
      LX2    = LX     + NX
      LY     = LX2    + NX2
      LY2    = LY     + NX
      LGSUB  = LY2    + NX
      LGSUB2 = LGSUB  + NN
      LGRD   = LGSUB2 + 0000
      LR     = LGRD   + MBS
      LRG    = LR     + NR
      LRG2   = LRG    + MAXS
      LXN    = LRG2   + MAXS
      MINZ   = LXN    + NB
      MINMPS = MINZ

*     Nonlinear objective.

      LGOBJ  = MINZ
      LGOBJ2 = LGOBJ  + NNOBJ
      MINZ   = LGOBJ2 + NNOBJ

*     Nonlinear constraints.

      IF (NDEN .EQ. 1) NJAC = NNCON*NNJAC
      NJAC   = MAX0( NJAC, 1 )
      LFCON  = MINZ
      LFCON2 = LFCON  + NNCON
      LFDIF  = LFCON2 + NNCON
      LFDIF2 = LFDIF  + NNCON
      LFOLD  = LFDIF2 + 0000
      LBLSLK = LFOLD  + NNCON
      LBUSLK = LBLSLK + NNCON
      LXLAM  = LBUSLK + NNCON
      LRHS   = LXLAM  + NNCON
      LGCON  = LRHS   + NNCON
      LGCON2 = LGCON  + NJAC
      LXDIF  = LGCON2 + NJAC
      LXOLD  = LXDIF  + NN
      MINZ   = LXOLD  + NN

*     Truncated CG. (Ignored in this version of MINOS.)

***** LENCG  = 0
```

```
**** IF (MODTCG .GE. 2) LENCG = MAXS
**** LCG1  = MINZ
**** LCG2  = LCG1  + LENCG
**** LCG3  = LCG2  + LENCG
**** LCG4  = LCG3  + LENCG
**** MINZ  = LCG4  + LENCG

*    Work arrays that can be overwritten by the names (see below)
*    in between CYCLES.

     LBBL  = MINZ
     LBBU  = LBBL  + MBS
     LGRD2 = LBBU  + MBS
     MINZ1 = LGRD2 + MBS

*    Arrays for names, for solution output.
*    These share space with the work arrays above.

     NCOLI  = 1 + N/NWORDI
     NROWI  = 1 + M/NWORDI
     LCNAM1 = MINZ
     LRNAM1 = LCNAM1 + NCOLI
     LCNAM2 = LRNAM1 + NROWI
     LRNAM2 = LCNAM2 + NCOLI
     MINZ2  = LRNAM2 + NROWI
     MINZ   = MAX0( MINZ1, MINZ2 )

*    Array for final solution values.
*    NOTE -- in this version, XN has length NB = N + M.
*    We can now define LXS to be the same as LXN.
*    LFREE points to the beginning of the LU factorization --
*    this is the beginning of free space after a SOLVE,
*    or between cycles if the LU is allowed to be overwritten.

     LXS   = LXN
     LXL   = LXS   + N
     LFREE = MINZ

*    Arrays for the basis factorization routines.

     CALL M2BMAP( MODE, M, N, NE, MINZ, MAXZ, NGUESS )

*    Temporary arrays for MPS input.
*    These share the storage used by  M2BMAP.

     LENH   = MAX0( 3*NROWI, 100 )
     IF (MODE .EQ. 3) LENH = 0
     LNAM1C = MINMPS
     LNAM1R = LNAM1C + NCOLI
     LNAM2C = LNAM1R + NROWI
     LNAM2R = LNAM2C + NCOLI
     LKEYNM = LNAM2R + NROWI
     MINMPS = LKEYNM + LENH

     IF (MODE .EQ. 1) MINCOR = MAX0( MINMPS, NGUESS, LFREE )
     IF (MODE .EQ. 2) MINCOR = MINMPS
     IF (MODE .EQ. 3) MINCOR = LFREE
```

```
*    End of M2CORE
     END
```

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	August 1990	Technical Report

4. TITLE AND SUBTITLE
Modifying MINOS for Solving the Dual of a Linear Program

5. FUNDING NUMBERS
DE-FG03-87ER25028
N00014-89-J-1659

6. AUTHOR(S)
Eithan Schweitzer

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Department of Operations Research - SOL
Stanford University
Stanford, CA 94305-4022

8. PERFORMING ORGANIZATION REPORT NUMBER
1111MA

9. SPONSORING MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Office of Naval Research - Department of the Navy
800 N. Quincy Street
Arlington, VA 22217

Office of Energy Research
U.S. Department of Energy
Washington, DC 20585

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
SOL 90-11

12a. DISTRIBUTION AVAILABILITY STATEMENT
UNLIMITED

12b. DISTRIBUTION CODE
UL

13. ABSTRACT *(Maximum 200 words)*

SOL 90-11: Modifying MINOS for Solving the Dual of a Linear Program, Eithan Schweitzer (August 1990, 48 pp.).

In solving large-scale linear programs by Benders' decomposition, it can be advantageous to solve the master and the sub problems via their dual problems. In this report I describe the changes I have made in one of the MINOS files, so MINOS could transform a given primal linear program to its dual, solve the dual and in addition, write an MPS file that contains the dual problem.

14. SUBJECT TERMS
Linear programming software; MINOS modification; solving dual.

15. NUMBER OF PAGES
48 pp.

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED			SAR